(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 2002157291 A

(43) Date of publication of application: 31.05.02

(51) Int. Cl G06F 17/50
H01L 21/82
H01L 27/04
H01L 21/822

(21) Application number: 2000349938

(22) Date of filing: 16.11.00

(71) Applicant: MATSUSHITA ELECTRIC IND CO LTD

(72) Inventor: TAKEMURA KAZUYOSHI

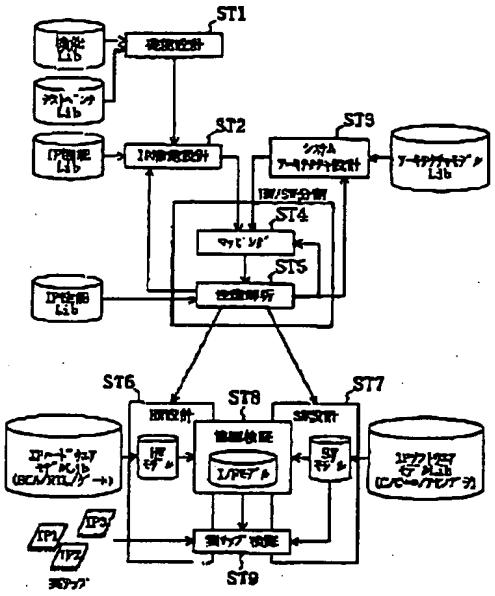
# (54) METHOD FOR DESIGNING SEMICONDUCTOR INTEGRATED CIRCUIT DEVICE

(57) Abstract:

PROBLEM TO BE SOLVED: To provide a method for designing a semiconductor integrated circuit device for selecting IP flexibly while securing the function of a system surely.

SOLUTION: After performing function design and IP function design, the architecture design of the system is performed. After mapping for assigning respective functions (function blocks) to the component of architecture, performance analysis is performed. Next, HW design, SW design, HW/SW cooperative verification, real chip verification are performed. By performing hierarchical development of an IP function at a function block level in the IP function design in particular, HW/SW division by each function block becomes possible and the system is optimized. Furthermore, dynamic analysis with switching of HW/SW by each function block becomes possible.





(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－157291
（P2002－157291A）
(43)公開日　平成14年５月31日(2002.5.31)

| (51)Int.Cl.' | 識別記号 | ＦＩ | テーマコード'（参考） |
|---|---|---|---|
| Ｇ０６Ｆ　17/50 | ６５４ | Ｇ０６Ｆ　17/50 | ６５４Ｍ　５Ｂ０４６ |
| | | | ６５４Ｇ　５Ｆ０３８ |
| | | | ６５４Ｋ　５Ｆ０６４ |
| | ６５２ | | ６５２Ｃ |
| | ６５８ | | ６５８Ｔ |

審査請求　有　請求項の数18　ＯＬ　（全 24 頁）　最終頁に続く

(21)出願番号　特願2000－349938(P2000－349938)

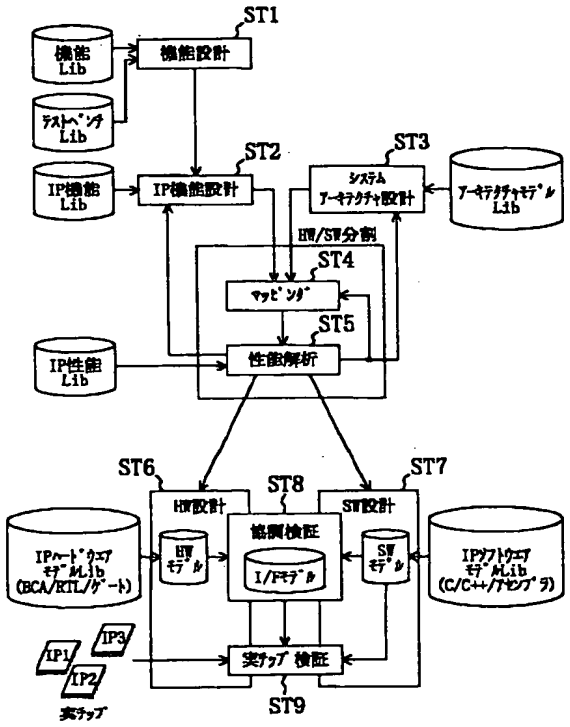
(22)出願日　平成12年11月16日(2000.11.16)

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者　竹村　和祥
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(74)代理人　100077931
弁理士　前田　弘　（外７名）

最終頁に続く

(54)【発明の名称】　半導体集積回路装置の設計方法

(57)【要約】
【課題】　システムの機能を確実に確保しながらＩＰの柔軟な選択を行なうための半導体集積回路装置の設計方法を提供する。
【解決手段】　機能設計と、ＩＰ機能設計を行なったあと、システムのアーキテクチャ設計を行なう。各機能（機能ブロック）をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、性能解析を行なう。次に、HW設計と、SW設計と、HW／SW協調検証と、実チップ検証とを行なう。特に、ＩＰ機能設計においてＩＰ機能を機能ブロックレベルに階層展開を行なうことにより、機能ブロックごとにHW／SW分割することが可能になり、より最適化されたシステムが得られる。また、機能ブロックごとにHW／SWの切り換えを伴う動的解析も可能になる。



1

【特許請求の範囲】

【請求項１】　共通の機能を有する複数のＩＰを、複数の機能について記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のＩＰから１つのＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）とを含む半導体集積回路装置の設計方法。

【請求項２】　請求項１記載の半導体集積回路装置の設計方法において、

上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のHWモデルとＳWモデルとに割り当てるように分割するステップ（ｅ）とをさらに含むことを特徴とする半導体集積回路装置の設計方法。

【請求項３】　請求項２記載の半導体集積回路装置の設計方法において、

上記ステップ（ｃ）の後、上記ＩＰの階層を機能ブロックレベルに展開するステップ（ｄ）をさらに含み、

上記ステップ（ｅ）では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のHWモデルとＳWモデルとに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項４】　各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、

ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、

上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、

上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のHW（ハードウェア）モデルとＳW（ソフトウェア）モデルとに割り当てるように分割するステップ（ｅ）とを含む半導体集積回路装置の設計方法。

【請求項５】　請求項４記載の半導体集積回路装置の設計方法において、

上記ステップ（ｅ）では、機能ブロックのHWの部分から面積の重複を考慮した静的解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項６】　請求項４記載の半導体集積回路装置の設計方法において、

上記ステップ（ｅ）の後、割り当てられた上記機能群構造中の各機能ブロックのHWモデルとＳWモデルとへの割り当てを決定するための動的性能解析を行なうステップ（ｆ）をさらに含むことを特徴とする半導体集積回路装置の設計方法。

2

【請求項７】　請求項６記載の半導体集積回路装置の設計方法において、

上記データベースには、HWの機能毎に当該HWと等価な機能を有するＳWモデルを格納しておくことを特徴とする半導体集積回路装置の設計方法。

【請求項８】　請求項７記載の半導体集積回路装置の設計方法において、

上記ステップ（ｆ）では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとＳWモデルとへの割り当てを切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項９】　請求項５記載の半導体集積回路装置の設計方法において、

上記ステップ（ｆ）では、ある機能ブロックをHWに割り当てたときのＣＰＵの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをＳWに切り換えることを特徴とする半導体集積回路装置の設計方法。

【請求項１０】　請求項５～９のうちいずれか１つに記載の半導体集積回路装置の設計方法において、

上記ステップ（ｅ）では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項１１】　複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、

上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）とを含み、

上記ステップ（ｂ）では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項１２】　請求項１１記載の半導体集積回路装置の設計方法において、

上記ステップ（ｂ）では、上記バスブリッジモデルが、バス幅が互いに異なる２つのバス間をバス幅及びデータ転送速度を調整するように接続していることを特徴とする半導体集積回路装置の設計方法。

【請求項１３】　複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、

上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）と、

上記仕様を満たすための機能群構造を構築するステップ（ｃ）と、

上記機能群構造の入力側及び出力側にテストベンチを付設するステップ（ｄ）と、

上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）とを含む半導体集積回路装置の設計方法。

【請求項１４】　請求項１３記載の半導体集積回路装置の設計方法において、

上記ステップ（ａ）では、上記複数のコンポーネントの中にＩ／Ｆモデルを含めておき、

上記ステップ（ｂ）では、上記コンポーネント群にＩ／Ｆモデルを含めるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。

【請求項１５】　請求項１４記載の半導体集積回路装置の設計方法において、

上記ステップ（ｅ）では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のＩ／Ｆモデルに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項１６】　請求項１３～１５のうちいずれか１つに記載の半導体集積回路装置の設計方法において、

上記ステップ（ａ）では、上記複数のコンポーネントの中にメモリ含めておき、

上記ステップ（ｂ）では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、

上記ステップ（ｅ）では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のメモリに割り当てることを特徴とする半導体集積回路装置の設計方法。

【請求項１７】　各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、

ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、

上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、

上記階層展開された各機能ブロックをアーキテクチャモデル中のコンポーネントに割り当てるステップ（ｅ）と、

上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャモデル中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ（ｆ）とを含む半導体集積回路装置の設計方法。

【請求項１８】　各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、

ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、

上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、

上記機能群中の各機能をデータベースのアーキテクチャ



モデル中のコンポーネントに割り当てるステップ（ｅ）と、

上記コンポーネントのバスアクセスのための優先度を設定するステップ（ｆ）と、

上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ（ｇ）とを含み、

上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算することを特徴とする半導体集積回路装置の設計方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数の機能を有する回路を複数個配置した半導体集積回路装置の設計方法に関する。

【０００２】

【従来の技術】近年、複数のＬＳＩを共通の基板上に形成したシステムＬＳＩという概念が提起されており、システムＬＳＩの設計手法としても各種の提案がなされている。特に、システムＬＳＩの利点は、ＤＲＡＭなどのメモリや、ロジックＬＳＩや、高周波回路などのアナログ回路を１つの半導体装置内に収納して、多種，多機能の半導体装置を極めて高集積化して実現することができることである。

【０００３】

【発明が解決しようとする課題】ところで、上記従来のシステムＬＳＩにおいては、システムＬＳＩを設計する際に、ＩＰ，コアと呼ばれる設計資産を利用することが多い。そのとき、各コアは個別に設計されたものなので、これらを統合したときには必ずしも円滑な動作が得られないことがある。しかし、従来のコア又はＩＰは、その中身がブラックボックスとなっているので、円滑な動作を確保するシステムを設計するためには多大の手間と時間とを要しているのが現実である。

【０００４】本発明の目的は、ＩＰの機能の同一性を図りつつＩＰの選択を柔軟に行なうための手段を講ずることにより、最適化が容易な半導体集積回路装置の設計方法の提供を図ることにある。

【０００５】

【課題を解決するための手段】本発明の第１の半導体集積回路装置の設計方法は、共通の機能を有する複数のＩＰを、複数の機能について記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のＩＰから１つのＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）とを含んでいる。

5

【０００６】この方法により、機能が互いに共通する複数のＩＰをＬｉｂに格納しておいて、機能ごとに適当なＩＰを選択することができるので、ＩＰを代えても機能の同一性が保持される。したがって、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【０００７】上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ（ｅ）とをさらに含むことにより、最適化された半導体集積回路装置を設計することができる。ができる。

【０００８】上記ステップ（ｃ）の後、上記ＩＰの階層を機能ブロックレベルに展開するステップ（ｄ）をさらに含み、上記ステップ（ｅ）では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のHWモデルとSWモデルとに割り当てることにより、より最適化された半導体集積回路装置を設計することができる。

【０００９】本発明の第２の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のHWモデルとSWモデルとに割り当てるように分割するステップ（ｅ）とを含んでいる。

【００１０】このように、ＩＰを階層展開してからHWモデルとSWモデルとに分割することにより、ＩＰ内をHWモデルとSWモデルとに区別してマッピングを行なうことが可能となり、ＩＰ毎にマッピングする場合に比べて、より最適化されたシステムの設計が可能になる。

【００１１】上記ステップ（ｅ）では、機能ブロックのHWの部分から面積の重複を考慮した静的解析を行なうことにより、面積の計算精度の悪化を防止することができる。

【００１２】上記ステップ（ｅ）の後、割り当てられた上記機能群構造中の各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換える動的性能解析を行なうステップ（ｆ）をさらに含むことにより、解析精度の向上を図ることができる。

【００１３】上記データベースには、HWの機能毎に当該HWと等価な機能を有するSWモデルを格納しておくことが好ましい。

【００１４】上記ステップ（ｆ）では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換えることができる。

6

【００１５】上記ステップ（ｆ）では、ある機能ブロックをHWモデルに割り当てたときのＣＰＵの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをSWに切り換えることもできる。

【００１６】記ステップ（ｅ）では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【００１７】本発明の第３の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）とを含み、上記ステップ（ｂ）では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築する方法である。

【００１８】このような複数のバスを設けることにより、ＩＰＯＳデバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。

【００１９】上記ステップ（ｂ）では、上記バスブリッジモデルが、バス幅が互いに異なる２つのバス間をバス幅及びデータ転送速度を調整するように接続していることにより、当該バスブリッジモデルに接続される２つのバスのバス幅やデータ転送速度が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。

【００２０】本発明の第４の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）と、上記仕様を満たすための機能群構造を構築するステップ（ｃ）と、上記機能群構造の入力側及び出力側にテストベンチを付設するステップ（ｄ）と、上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）とを含んでいる。

【００２１】この方法により、テストベンチをHWにマッピングすることが可能になり、性能解析などにテストベンチの入出力に要する時間などのパラメータを含めることができる。

【００２２】上記ステップ（ａ）では、上記複数のコンポーネントの中にＩ／Ｆモデルを含めておき、上記ステップ（ｂ）では、上記コンポーネント群にＩ／Ｆモデルを含めるように上記アーキテクチャを構築することにより、入出力の信号を取り扱うＩ／Ｆモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【００２３】上記ステップ（ｅ）では、上記入力側のテ

ストベンチを上記アーキテクチャ中のＩ／Ｆモデルに割り当てることにより、従来、単に入力信号として扱われていたテストベンチがＨＷであるＩ／Ｆモデルにマッピングされるので、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、Ｉ／Ｆが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【００２４】上記ステップ（ａ）では、上記複数のコンポーネントの中にメモリ含めておき、上記ステップ（ｂ）では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、上記ステップ（ｅ）では、上記出力側のテストベンチを上記アーキテクチャモデル中のメモリに割り当てることもできる。

【００２５】本発明の第５の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、上記階層展開された各機能ブロックをアーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）と、上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャ中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ（ｆ）とを含んでいる。

【００２６】このように機能ブロック間パステーブルを作成することにより、動的な性能解析を精度よく行なうことができる。

【００２７】本発明の第６の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ（ｅ）と、上記コンポーネントのバスアクセスのための優先度を設定するステップ（ｆ）と、上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ（ｇ）とを含み、上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算する方法である。

【００２８】優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００２９】

【発明の実施の形態】－ＩＰＯＳデバイスの基本的な構造－

ここで配線の断面積を大きく確保しつつ、多種，多機能のデバイスを内蔵した半導体装置を実現するための１つの手段として、配線層を有する半導体配線基板例えばシリコン配線基板（Ｓｕｐｅｒ－Ｓｕｂ）上に、各種デバイスを内蔵したチップＩＰを搭載する構成がある。その場合、各チップＩＰ内に設けられる回路（ＩＣ）は半導体装置の設計上ＩＰ(Intellectual Property)として扱うことができ、各種ＩＰを半導体配線基板上に貼り合わせたものと考えることができる。つまり、半導体デバイス全体は、“ＩＰ　Ｏｎ　Ｓｕｐｅｒ－Ｓｕｂ”であるので、シリコン配線基板とＩＰ群とを備えた半導体デバイス全体を“ＩＰＯＳデバイス”とする。

【００３０】図１（ａ），（ｂ），（ｃ）は、ＩＰ（チップＩＰ）群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるＩＰ群の例を示す平面図、及びシリコン配線基板の断面図である。図１（ａ），（ｂ）に示すように、シリコン配線基板１０上には各種ＩＰを搭載するための複数の領域が設けられており、、各領域には、例えば、Analog－ＩＰ，Logic －ＩＰ，ＣＰＵ－ＩＰ，Flash メモリーＩＰ，ＳＲＡＭ－ＩＰ，ＤＲＡＭ－ＩＰ，Ｉ／Ｏ－ＩＰなどの各種ＩＰ群がチップＩＰとして搭載可能となっている。図１（ｃ）に示すように、シリコン配線基板１０は、シリコン基板１１と、シリコン基板１０上に絶縁膜（図示せず）を挟んで設けられたグランドプレーン１２と、グランドプレーン１２の上に層間絶縁膜を挟んで設けられた第１配線層１３と、第１配線層１３の上に層間絶縁膜を挟んで設けられた第２配線層１４と、第２配線層１４の上にパッシベーション膜を挟んで設けられたパッド１５とを備えている。パッド１５，各配線層１３，１４及びグランドプレーン１２間は、それぞれコンタクト（図示せず）を介して所望の部位で互いに接続されている。そして、各ＩＰは、パッド１５上に貼り付けられて、各ＩＰが配線層１３，１４により互いにあるいはグランドプレーン１２に電気的に接続される構造となっている。

【００３１】シリコン配線基板１０内の配線層１３，１４の寸法の制約は緩やかであり、数μｍ幅の配線をも設けることができるので、以下のような効果がある。経験的に、今までの半導体集積回路装置の微細化が進展した過程において、もっとも配線としての特性が良好であった世代の寸法を有する配線を設けることが可能となる。また、配線の電気インピーダンスを低減することができる。

【００３２】そして、シリコン配線基板上のチップＩＰは多くの機能を有しているが、これに対しては２つの考え方がある。１つは、できるだけ多くの機能を使用する

という考え方であり、もう１つは、使用する機能を制限するという考え方である。つまり、ユーザの多様な要求に応えるためには、多種の使用方法に対応できる構成を有していることが好ましいが、反面、実際の使用に際してはいずれかの機能に限定する必要がある。つまり、この２つの相反する要求を満足させることにより、ユーザの多様な要求に応えつつ、少品種，大量生産に適したＩＰＯＳデバイスを構築することができるのである。

【００３３】したがって、後に選択，制限，切り換え，設定などの処理を柔軟に行なうことを前提として、つまり、機能に関する処理の柔軟性を保持しつつ，多種の機能をハードウェア（ＨＷ），ソフトウェア（ＳＷ）に分けて各ＩＰ（チップＩＰ）に組み込むことが重要となる。

【００３４】そこで、以下の実施形態においては、ＩＰＯＳデバイスを例とした大規模なシステム（以下、「ＩＰＯＳシステム」という）に適したＨＷ／ＳＷ協調設計について説明する。ただし、本発明は、必ずしもＩＰＯＳシステムだけでなく、従来のいわゆるシステムＬＳＩなどの大規模な集積回路システムの設計にも適用が可能である。

【００３５】（実施形態）

－ＨＷ／ＳＷ協調設計全体の概略－

図２は、本発明の実施形態におけるＨＷ／ＳＷ協調設計の概略的な手順を示すフロー図である。

【００３６】ステップＳＴ１で、機能ライブラリ（機能Ｌｉｂ）やテストベンチを用いて機能設計を行ない、ステップＳＴ２で、ＩＰ機能Ｌｉｂを用いたＩＰ機能設計を行なう。このステップＳＴ１，ＳＴ２では、半導体デバイスに搭載される各ＩＰに要求される機能が例えばＣ言語により記述される。例えばＪＰＥＧの場合には、信号処理（アルゴリズム）だけが記載されているような状態である。

【００３７】次に、ステップＳＴ３で、アーキテクチャモデルＬｉｂを用いてシステムのアーキテクチャ設計を行なう。システムのアーキテクチャモデルＬｉｂには、アーキテクチャモデルがシステムの設計に必要なＨＷ，ＳＷモデルとして存在しているので、このステップＳＴ３では、各モデルをバスに接続してシステム全体の概略的な構造を作り上げる。

【００３８】次に、ステップＳＴ４で、各機能（機能ブロック）をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、ステップＳＴ５で、性能Ｌｉｂを用いて性能解析を行ない、その結果、不具合があればステップＳＴ３のアーキテクチャ設計，又はステップＳＴ４のマッピングに戻り、性能が満足されるまでこれらのステップを繰り返すループ処理を行なう。

【００３９】次に、ステップＳＴ６，ＳＴ７で、それぞれＩＰハードウェアモデルＬｉｂを用いたＨＷ設計と、例えばＩＰソフトウェアモデルＬｉｂを用いたＳＷ設計とを個別に行なってから、ＨＷ及びＳＷが詳細化されてくると、ステップＳＴ８で、Ｉ／Ｆモデルを用いたＨＷ／ＳＷ協調検証を行なう。その後、ステップＳＴ９で、実チップを用いて実チップの機能を確認するための実チップ検証を行なう。

【００４０】－機能設計及びＩＰ機能設計－

図３は、機能設計を行なうステップＳＴ１と、ＩＰ機能設計を行なうステップＳＴ２との詳細な処理の内容を説明するための部分フロー図である。

【００４１】ステップＳＴ１においては、以下の手順により、機能設計を行なう。機能Ｌｉｂには、ｆ１，ｆ２，ｆ３，ｆ４などの種々の機能（ｆｕｎｃｔｉｏｎ）が格納されている。この機能ｆ１，ｆ２，ｆ３，ｆ４とは、例えば画像処理におけるＭＰＥＧＫＬ，ＪＰＥＧ，Blue-toothなどの機能である。また、テストベンチＬｉｂには、ｔｂ１，ｔｂ２，ｔｂ３，ｔｂ４などの種々のテストベンチが格納されている。このテストベンチｔｂ１，ｔｂ２，ｔｂ３，ｔｂ４は、機能設計が適正か否かを判断するための与えられるデータである。例えば、図３のステップＳＴ１中に示すように、機能ｆ１，ｆ２，ｆ３を組み合わせたデザインがある場合には、入力テストベンチとして、テストベンチからｔｂ１を取り出して、これを入力データとして与える。また、出力データとして例えば画像出力のテストベンチｔｂ２を与える。その結果、ここで作り上げたデザインが適正かどうかがある程度検証される。

【００４２】次に、ステップＳＴ２におけるＩＰ機能設計は以下の手順で行なわれる。ＩＰ機能Ｌｉｂには、各機能ｆ１，ｆ２，ｆ３，ｆ４ごとに、この機能を実現するためのいくつかのＩＰが格納されている。例えば、機能ｆ１に対してｆ１－ｉｐ１～ｆ１－ｉｐ４があり、機能ｆ２に対してｆ２－ｉｐ１～ｆ２－ｉｐ３があり、機能ｆ３に対してｆ３－ｉｐ１，ｆ３－ｉｐ２がある。そこで、おおよその見当をつけて、デザイン中の機能ｆ１，ｆ２，ｆ３に対して用いるＩＰを指定する。例えば、機能ｆ１に対してはｆ１－ｉｐ４を、機能ｆ２に対してはｆ２－ｉｐ１を、機能ｆ３に対してはｆ３－ｉｐ２を用いるというようにパラメータを指定する。

【００４３】次に、ステップＳＴ１で作り上げられたデザイン中の各機能ｆ１，ｆ２，ｆ３に、それぞれｆ１－ｉｐ４，ｆ２－ｉｐ１，ｆ３－ｉｐ２を当てはめる。つまり、ＩＰ機能の選択を行なう。この段階では、選び出された各ＩＰはいずれもステップＳＴ１における機能ｆ１，ｆ２，ｆ３を有しているので、ステップＳＴ１で検証した結果と同じ結果が得られる。

【００４４】このように、機能が互いに共通する複数のＩＰをＬｉｂに格納しておいて、機能ごとに適当なＩＰを選択することにより、ＩＰを代えても機能の同一性が保持されるので、従来大きかった設計変更の確率をできるだけ小さくすることができる。

11

【００４５】次に、ＩＰ機能をさらに詳細に個々の機能に分けるＩＰ機能の階層展開を行なう。例えば、ｆ１－ｉｐ４について階層展開をすると、ｆ１－ｉｐ４－Ａ（例えば逆コサイン変換）を行なってからｆ１－ｉｐ４－Ｂ（例えば量子化）を行ない、その後ｆ１－ｉｐ４－Ｃ（例えばハフマン符号化）を行なうという機能になっていたとする。つまり、ＩＰであるｆ１－ｉｐ４を機能ブロックｆ１－ｉｐ４－Ａ，ｆ１－ｉｐ４－Ｂ，ｆ１－ｉｐ４－Ｃに展開するのである。同様に、ｆ２－ｉｐ１，ｆ３－ｉｐ２についても階層展開を行なって、ｆ２－ｉｐ１についてはｆ２－ｉｐ１－Ａという機能ブロックが、ｆ３－ｉｐ２についてはｆ３－ｉｐ２－Ａ，ｆ３－ｉｐ２－Ｂという機能ブロックがそれぞれ得られたとする。そして、この階層展開された各機能ブロックをデザイン中の各機能ｆ１，ｆ２，ｆ３に置換する。この階層展開されたＩＰの中身をＨＷ／ＳＷ分割するのが本発明の１つの特徴である。

【００４６】－アーキテクチャ設計－

次に、図４は、ステップＳＴ３における処理の内容を説明するための部分フロー図である。同図に示すように、アーキテクチャモデルＬｉｂには、ＣＰＵ，メモリ，ＡＳＩＣ（ロジック回路），ＢＵＳ，バスブリッジモデル，Ｉ／Ｆモデルなどのコンポーネント（ｃ）についての多くの種類が格納されている。そこで、このコンポーネントごとに各種のコンポーネントパラメータをもっているので、コンポーネントパラメータの値を与える。つまり、ＣＰＵについては動作周波数，キャッシュヒット率があり、ＤＲＡＭについては動作周波数，ビット幅があり、ＡＳＩＣについては動作周波数，ビット幅があり、ＢＵＳについてはバスクロック，バス幅，転送レート（ｂｕｓ），アービトレーション方法があり、バスブリッジモデルについてはバスブリッジ間遅延があり、Ｉ／ＦモデルとしてＵＡＲＴを選択したときには入出力転送レート（ｂｐｓ），入出力ビット幅，動作周波数，ビット幅がある。そして、各コンポーネントパラメータに具体的な値を入れる。そして、このパラメータ指定を行なってから、例えば図４に示すように、ＣＰＵ１（ｃ１），ＤＲＡＭ１（ｃ２），ＡＳＩＣ１（ｃ３），ＡＳＩＣ２（ｃ６）及びＵＡＲＴ（ｃ５）をＢＵＳ１，ＢＵＳ２及びバスブリッジ１（ｃ４）を介して接続した構造を有するアーキテクチャを作り上げる。

【００４７】ここで、このアーキテクチャの特徴は、複数のバス（ＢＵＳ１，ＢＵＳ２）を備え、ＢＵＳ１－ＢＵＳ２間を接続するバスブリッジ１（ｃ４）をコンポーネントとして設けている点である。バスは３つ以上あってもよく、このような複数のバスを設けることにより、半導体デバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。複数のバスの例としては、標準バスに低速バス，高速バスなどを別途設けたものが考えられる。そして、バスブリッジ

12

モデルを設けることにより、当該バスブリッジモデルに接続される２つのバスのビット幅が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。バスブリッジモデルの例としては、１６ビット幅のＢＵＳ（例えばバス１）のデータを８ビットずつ別のＢＵＳ（例えばＢＵＳ２）に送り込むバッファがある。また、アーキテクチャにおいて、Ｉ／ＦモデルをＡＳＩＣとは別に設けたことにより、入出力の信号を取り扱うＩ／Ｆモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【００４８】－マッピング－

次に、図５は、ステップＳＴ４，ＳＴ５におけるマッピングと性能解析の概略とを説明するための部分フロー図である。

【００４９】同図に示すように、ステップＳＴ４においては、ステップＳＴ２のＩＰ機能設計によって得られたデザイン中の階層展開されたＩＰを、ＨＷとＳＷとに割り当てる処理であるマッピングを行なう。例えば、テストベンチｔｂ１は、ＨＷとしてＩ／ＦモデルであるＵＡＲＴに割り当てられる。また、テストベンチｔｂ２は、ＨＷとしてＤＲＡＭ１に割り当てられる。また、階層展開されているｆ１－ｉｐ４中の機能ブロックｆ１－ｉｐ４－Ａ（ｂｌ１）はＨＷとしてＡＳＩＣ１に、機能ブロックｆ１－ｉｐ４－Ｂ（ｂｌ２），機能ブロックｆ１－ｉｐ４－Ｃ（ｂｌ３）はＳＷとしてＣＰＵ１に、ｆ２－ｉｐ１中の機能ブロックｆ２－ｉｐ１－Ａ（ｂｌ４）はＳＷとしてＣＰＵ１に、ｆ２－ｉｐ２中の機能ブロックｆ２－ｉｐ２－Ａ（ｂｌ５），機能ブロックｆ２－ｉｐ２－Ａ（ｂｌ６）はすべてＨＷとしてＡＳＩＣ２にそれぞれ割り当てられる。

【００５０】ここで、このマッピングの特徴は、テストベンチをＩ／Ｆモデルにマッピングしたことである。従来、単に入力信号として扱われていたテストベンチをＨＷであるＩ／Ｆモデルにマッピングすることにより、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、Ｉ／Ｆモデルが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【００５１】また、このように、１つのＩＰをＨＷとＳＷとにわけてマッピングができるようになったのは、ＩＰ機能設計において、上述のようにＩＰを機能ブロックのレベルまで階層展開したからである。そして、マッピングに応じて、図５の左下に示すように、各機能とデザイン中のコンポーネントとのマッピング対応表が機能番号（ａ１～ａｎ）とともに作成される。また、例えば機能ブロックｆ１－ｉｐ４－Ａ→機能ブロックｆ１－ｉｐ４－Ｂというパスは、ＡＳＩＣ１→ＢＵＳ１→（ＤＲＡＭ１→）ＣＰＵ１というデータのパスとなり、機能ブロックｆ１－ｉｐ４－Ｂ→機能ブロックｆ１－ｉｐ４－Ｃというパスは、ＣＰＵ１→（ＤＲＡＭ１→）ＣＰＵ２と

13

いうデータのパスとなることが、ブロック間パステーブルとして作成され、設計装置の記憶装置（図示せず）に格納される。なお、括弧内のコンポーネントはその部分が経由される場合もあり、経由されない場合もあることを示している。

【００５２】ここで、このように機能ブロック間パステーブルを作成することにより、後述するような動的な性能解析を精度よく行なうことができる。

【００５３】次に、このようなマッピング結果を用いてステップＳＴ５における性能解析を行なう。このステップにおいては、静的解析と動的解析とを行なってもよいし、静的解析及び動的解析のうちいずれか一方のみを行なってもよい。

14

【００５４】－性能解析（静的解析）－

性能解析のうち静的解析は、例えば所望のパラメータを変数としてシステム全体のコスト関数（Cost-tot）を定義して、このコスト関数（Cost-tot）を求める。そして、コスト関数（Cost－tot）が最小又はある上限値以下になるかを目標として設計を進める。このコスト関数（Cost-tot）の一例として、例えば下記式（１）

$$\begin{aligned}Cost\text{-}tot = {} & k1\ (Speed_{HW\text{-}tot} + Speed_{SW\text{-}tot}) \\ & k2\ (Area_{HW\text{-}tot} + Area_{SW\text{-}tot}) \\ & k3\ (Power_{HW\text{-}tot} + Power_{SW\text{-}tot}) \end{aligned} \quad (1)$$

がある。ここで、ｋ１，ｋ２，ｋ３は重み係数であり、$Speed_{HW\text{-}tot}$はＨＷ全体の処理速度であり、$Speed_{SW\text{-}tot}$はＳＷ全体の処理速度であり、$Area_{HW\text{-}tot}$はＨＷ全体の面積であり、$Area_{SW\text{-}tot}$はＳＷ全体の面積であり、$Power_{HW\text{-}tot}$はＨＷ全体の消費電力であり、$Power_{SW\text{-}tot}$はＳＷ全体の消費電力である。つまり、この例では、パラメータである処理速度ができるだけ大きく、面積ができるだけ小さく、消費電力ができるだけ小さくなるようにシステムを構築したいと考えていることにな※る。そこで、コスト関数（Cost-tot）は小さいほどよいとする。例えば、各種のマッピングＭＡＰ１，ＭＡＰ２，ＭＡＰ３を行なった結果、コスト関数（Cost-tot）が図６に示す値になったとする。このときには、もっとも値が小さいマッピングＭＡＰ１が最適であると判断する。

【００５５】ここで、例えば処理速度に関する計算は、マッピング結果を用いて、下記式（２）

$$\begin{aligned}Speed_{HW\text{-}tot}\ (f_{b11} \sim f_{b16},\ f_{bus},\ R_{bus},\ P_{b11} \sim P_{b16}) \\ = Speed_{HW\text{-}b11}\ (f_{c6},\ f_{bus},\ R_{bus},\ R_{c6}) + \\ \cdots\cdots + Speed_{HW\text{-}b16}\ (f_{c6},\ f_{bus},\ R_{bus},\ P_{c6}) \end{aligned} \quad (2)$$

のように行なわれる。ここで、式（２）において、図５に示すように、マッピングしたときの機能ブロック番号をｂｌ１～ｂｌ６とし、コンポーネント番号をｃ１～ｃ６とする。そして、ｆは動作周波数であり、Ｒはバスの転送レートであり、Ｐは機能のパスに対する優先順位（優先度）（０～１の値）を表している。この優先順位★

$$\begin{aligned}Speed_{HW\text{-}b11}\ (f_{c6},\ f_{bus},\ R_{bus},\ P_{c6}) \\ = 1 / (f_{c6} \times f_{bus} \times R_{bus} \times P_{c6}) \end{aligned} \quad (3)$$

のように求められる。つまり、処理速度はできる限り小さい方が好ましいことから、動作周波数ｆ，転送レートＲ，優先順位Ｐの逆数からコスト関数（Cost-tot）のうち処理速度Ｓｐｅｅｄの成分を計算する。☆

$$\begin{aligned}Speed_{SW\text{-}tot}\ (St_{CPU\text{-}c1} \sim St_{CPU\text{-}c6},\ f_{CPU}) \\ = Speed_{SW\text{-}b11}\ (St_{CPU\text{-}c1},\ f_{CPU}) + \\ \cdots + Speed_{SW\text{-}b16}\ (St_{CPU\text{-}c6},\ f_{CPU}) \end{aligned} \quad (4)$$

により求められる。この計算の考え方は以下の通りである。機能ブロックをｆｕｎｃｔｉｏｎによって表すと、処理はＣ言語によって記述することができ、Ｃ言語で記述すればアセンブリ言語にコンパイルすることが可能となる。したがって、アセンブラのステップ数で処理速度◆

$$Speed_{SW\text{-}b11}\ (St_{CPU\text{-}c1},\ f_{CPU}) = St_{CPU\text{-}c1} \times f_{CPU} \quad (5)$$

のように求められる。このステップ数が小さいほど速度が大きいので、式（５）の数値がなるべく小さいことが望ましいことになる。

$$Area_{HW\text{-}tot} = Area_{HW\text{-}f1\text{-}ip4}$$

★Ｐは、バスに接続されるすべてのコンポーネントの優先度を合計すると１になるように設定されている。

【００５６】すると、式（２）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の処理速度は、下記式（３）

☆【００５７】次に、式（１）中のＳＷについての処理速度成分$Speed_{SW\text{-}tot}$は、マッピングのブロック間パステーブルに基づいて、下記式（４）

◆を演算することができるからである。

【００５８】ここで、すると、式（４）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の処理速度は、下記式（５）

【００５９】次に、面積に関する計算は、マッピング結果を用いて、下記式（６）

15

$$+\mathrm{Area}_{HW\text{-}f2\text{-}ip1}+\mathrm{Area}_{HW\text{-}f3\text{-}ip2}$$

のように行なわれる。つまり、ＩＰ機能を選択した後でＩＰ機能の階層展開を行なう前に使用される各ＩＰ（ｆ１－ｉｐ４，ｆ２－ｉｐ１，ｆ３－ｉｐ２）の情報があるので、これからHW部分の面積を求めることができる。

【００６０】次に、機能ブロックをコンパイルすると、＊

$$\mathrm{Area}_{SW\text{-}tot}=(\mathrm{Msize}_{SW\text{-}b11}+\cdots+\mathrm{Msize}_{SW\text{-}b16})\times\mathrm{Area}_{DRAM}\quad(7)$$

により、求めることができる。

【００６１】このように、ＩＰ内が機能ブロックに分割されたことにより、機能ブロックのＳＷの部分からメモリサイズを計算すると、各機能ブロック間で面積を重複※

$$\mathrm{Power}_{HW\text{-}tot}=\mathrm{Power}_{HW\text{-}b1}+\cdots+\mathrm{Power}_{HW\text{-}b16}\quad(8)$$

のように行なわれる。すると、式（８）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の消費電力は、下★

$$\mathrm{Power}_{HW\text{-}b11}(f_{cs},\ V_{cs})=\alpha_{b11}\times f_{cs}\times(V_{cs})^{2}\times P_{cs}\quad(9)$$

により、表される。ただし、αは機能ブロックｂｌ１のパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$f_{cs}$はコンポーネントｃ３（ＡＳＩＣ１）の動作周波数、$V_{cs}$はコンポーネントｃ３（ＡＳＩＣ１）の電圧、$P_{cs}$はコンポーネントｃ３（ＡＳＩＣ１）の優先順位（優先☆度）を示している。この優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００６３】また、ＳＷ部分の消費電力に関する計算は、下記式（１０）

$$\mathrm{Power}_{SW\text{-}tot}=\mathrm{Power}_{SW\text{-}CPU\text{-}tot}+\mathrm{Power}_{SW\text{-}MEM\text{-}tot}\quad(10)$$

により求められる。式（１０）の第１項は、

$$\mathrm{Power}_{SW\text{-}CPU\text{-}tot}=\alpha_{CPU}\times f_{CPU}\times \mathrm{Load}_{CPU}\quad(11)$$

となる。ここで、$\alpha_{CPU}$はＣＰＵのパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$\mathrm{Load}_{CPU}$はＣＰＵロード（０～１）であって、「ＳＷにマッピングされた機能ブロックのメモリサイズの合計（ｂｉｔ）」を「すべての◆

$$\mathrm{Power}_{SW\text{-}MEM\text{-}tot}=\alpha_{DRAM}\times f_{DRAM}\times(V_{DRAM})^{2}\times P_{DRAM}\quad(12)$$

となる。$\alpha_{DRAM}$はＤＲＡＭのパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$f_{DRAM}$はＤＲＡＭの動作周波数（Ｈｚ）、$V_{DRAM}$はＤＲＡＭの電圧、$P_{DRAM}$はＤＲＡＭの優先順位を示している。この優先度を用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００６５】そして、以上の計算を行なって上で、式（１）から最終的に決定されるコスト関数（Cost-tot）が最小になるか、下限値以下になったときに静的解析に関する限り、設計されたシステムが適切であると判断する。

【００６６】－性能解析（動的解析）－

動的解析においては、面積Area については静的解析と同じであるので、処理速度Ｓｐｅｅｄと消費電力Ｐｏｗｅｒのみを求める。動的解析においては、設計システムに実際にデータを入力させて動作させるシミュレーションを行なう。したがって、実行時間（Ｔｉｍｅ （ｓｅｃ））や、各機能ブロックの入出力データ、トグル率（５）がシミュレーションから得られる。

【００６７】そこで、動的解析においては、基本的に

16

（６）

＊アセンブラでメモリサイズがわかるので、ＳＷのメモリの部分をすべて加算しておいて、それにエリアＤＲＡＭというデータベースを用いてＤＲＡＭの１ビット当たりの面積（単価）を計算する。それに、メモリサイズを乗じると、下記式（７）

※して計算する場合があるので、これを考慮して面積の計算精度の悪化を防止するのである。

【００６２】次に、HW部分の消費電力に関する計算は、マッピング結果を用いて、下記式（８）

★記式（９）

（９）

☆度）を示している。この優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００６３】また、ＳＷ部分の消費電力に関する計算は、下記式（１０）

（１０）

（１１）

◆機能ブロックをＳＷにマッピングしたときのメモリサイズの合計（ｂｉｔ）」で除したものである。

【００６４】一方、式（１０）の第２項は、

（１２）

は、この得られた要素を上記式（１）～（１２）に代入することにより、コスト関数（Cost-tot）の値を求めることができる。その際、式（１）における（$\mathrm{Speed}_{HW\text{-}tot}+\mathrm{Speed}_{SW\text{-}tot}$）は、シミュレーションで得られた時間Ｔｉｍｅに置き換えることができ、式（９）における各コンポーネントの優先順位Ｐはシミュレーションで得られた各コンポーネントのトグル数で置き換えることができる。そして、式（１１）におけるＣＰＵロードＬｏａｄ$_{CPU}$は、実際のシミュレーションから求まる平均ロード（５）を用いる。

【００６８】さらに、式（１２）におけるＤＲＡＭの優先順位Ｐ（優先度）はシミュレーションで得られたＤＲＡＭのトグル数で置き換えることができる。

【００６９】－ＨＷ／ＳＷ分割（静的分割）－

次に、ステップＳＴ４，ＳＴ５と共に行なわれるＩＰ内のHW／ＳＷ分割について説明する。HW／ＳＷには静的分割と動的分割とがあるが、まず、静的な分割について説明する。

【００７０】図７は、静的なHW／ＳＷ分割を説明するための図である。同図に示すように、半導体デバイスな

17

どのシステムに利用されるハードＩＰには、機能Ａ，機能Ｂ，機能Ｃ，機能Ｄがあり、かつ、これらの機能のＯＮ・ＯＦＦを設定するための設定レジスタがある。ただし、ＨＷ／ＳＷ分割における「機能」とは本実施形態における「機能ブロック」を意味している。一方、ＩＰ性能Ｌｉｂには、各機能Ａ，機能Ｂ，機能Ｃ，機能Ｄについてのモデルが例えばＣ／Ｃ＋＋言語によって記述されて格納いる。このとき、各機能をＨＷとＳＷとによって動作させた結果、１つの機能を変更，修正，削除する必要が生じる場合がある。ここで、例えば、設定レジスタ内の機能ＤをＯＦＦにすることで、例えば機能Ｄへのクロックの供給を停止するなどの制御を行なって、機能Ｄが動作しないようにすることができる。これにより、無駄な電力の消費を抑制して消費電力の低減を図ることができる。また、各機能Ａ～Ｄ間の接続を変更することもできる。

【００７１】さらに、例えば１つの機能ＤをＨＷからＩＰ性能Ｌｉｂに用意されたＳＷモデルに切り換えたり、用意しているＬｉｂ内のＳＷモデルを修正することもできる。つまり、いずれかのＬｉｂ内に各機能と等価なＳＷモデルを準備しておくことにより、１つのＩＰ内をＨＷ／ＳＷ分割することができ、ハードＩＰの機能が不足したときにも設計を進めることができる。また、ＳＷの機能を修正，追加することにより、システムの変更に柔軟に対応することができる。

【００７２】－動的なＨＷ／ＳＷ分割－

次に、動的なＨＷ／ＳＷ分割について説明する。図７に示すＨＷ／ＳＷ分割は、ＨＷ，ＳＷの設定を行なった後は、原則としてその後の修正は行なわない。それに対し、動的なＨＷ／ＳＷ分割，つまりＨＷ／ＳＷ動作の切り換えにおいては、機能（機能ブロック）が動作しながら柔軟に変更，追加，修正を受けることができる。このような動的なＨＷ／ＳＷ分割の例として、以下、第１～第３の具体例について説明する。

【００７３】１．第１の具体例

第１の具体例では、図７に示すＩＰにおいて、各機能のＯＮ，ＯＦＦをＳＷによって制御し、かつ、停止させる場合はＩＰ性能Ｌｉｂから同等のＳＷモデルを呼びだしてこれを用いる。設定レジスタへの制御信号により、各機能を動作させながらＨＷ／ＳＷの切り換えを行なうことができる。

【００７４】２．第２の具体例

図８（ａ），（ｂ）は、動的なＨＷ／ＳＷの切り換えにおける消費電力Ｐｏｗｅｒのタイムチャート図である。また、図９は、このＨＷ／ＳＷの切り換え制御のためのプログラムの例をＣ言語で示す図である。図９に示すように、設定レジスタの制御により、ハードＩＰの機能Ｃを動作させた場合に、図８（ａ）に示すように、ＩＰの消費電力Ｐｏｗｅｒが消費電力の上限値ＭＡＸｐｏｗｅｒ を越える時があるとする。動的解析からどの機能が動作している

18

かが時間軸に応じてわかるので、あるレジスタに各機能の消費電力の合計を計算しながら書き込んでおいて、その値が上限値ＭＡＸｐｏｗｅｒ を超えるときを検出すればよい。そして、ＩＰの消費電力Ｐｏｗｅｒが消費電力の上限値ＭＡＸｐｏｗｅｒ を越えると、図９に示すように、設定レジスタの制御によりハードＩＰの機能ＣをＯＦＦにして、代わりにＳＷモデルからｆｕｎｃ－Ｃを呼び出してこちらを動作させる。これにより、図８（ｂ）に示すように、消費電力Ｐｏｗｅｒが上限値ＭＡＸｐｏｗｅｒ を越えることがなくなり、所望の制約条件の下での作動を確保することができる。つまり、この例では、当該ＩＰの機能Ａ，Ｃが動作する期間には機能ＣがＳＷによって実現にされるが、機能Ａ，Ｂ，Ｃが動作する期間には機能ＣがＨＷによって実現されるうに設計することになる。

【００７５】３．第３の具体例

図１０（ａ），（ｂ）は、動的なＨＷ／ＳＷの切り換えにおける処理速度Ｓｐｅｅｄの消費電力依存性を示す図である。また、図１１は、このＨＷ／ＳＷの切り換え制御のためのプログラムの例をＣ言語で示す図である。図１１に示すように、（注：不等号の向きが逆のように思われましたので変更いたしましたが、どうでしょうか）ＣＰＵの設定レジスタの制御により、ＣＰＵのロード（％）がある値ＩＤＬＥよりも大きいときには機能ＣをＳＷにより実行し、ＣＰＵのロード（％）がある値ＩＤＬＥ以下のときに機能ＣをＨＷにより実行するよう制御する。つまり、機能Ｃを常にＨＷで実行すると、図１０（ａ）に示すように処理時間Ｔｉｍｅ が上限値ＭＡＸｔｉｍｅを越える事態が発生する。そこで、レジスタにＣＰＵのロード（％）を検知しながら書き込んでおいて、ＣＰＵのロード（％）が下限値ＩＤＬＥを越えるときを検出しながら、ＣＰＵのロード（％）がある値ＩＤＬＥ以下のときには機能ＣをＨＷにより実行する。そして、ＣＰＵのロード（％）が下限値ＩＤＬＥを越えると、設定レジスタの制御によりハードＩＰの機能ＣをＯＦＦにして、代わりにＳＷモデルからｆｕｎｃ－Ｃを呼び出してＳＷを動作させる。このような制御を行なうことにより、図１０（ｂ）に示すように、処理時間Ｔｉｍｅ が上限値ＭＡＸｔｉｍｅを越えることがなくなり、所望の制約条件の下での作動を確保することができる。

【００７６】－ＨＷ／ＳＷ協調検証－

次に、ステップＳＴ８の協調検証について説明する。上述のステップＳＴ４のマッピング処理，ステップＳＴ５の性能解析処理に基づいて最終的に決定されたＨＷ，ＳＷの構成に従い、ステップＳＴ６，ＳＴ７で、ＨＷ設計とＳＷ設計とを行なって、ＨＷモデルとＳＷモデルとをそれぞれ作成する。ＨＷモデルを作成する際にはＩＰハードウェアモデルＬｉｂ（ＢＣＡ，ＲＴＬ又はゲートレベルで記述）を利用し、ＳＷモデルを作成する際にはＩＰソフトウェアモデルＬｉｂ（Ｃ言語，Ｃ＋＋言語又はアセンブラで記述）を利用する。

19

【００７７】次に、ステップＳＴ８では、ステップＳＴ６，ＳＴ７で作成したHWモデル及びSWモデルを利用して、HW／SW協調検証を行なうが、そのためにはHW－SW間のＩ／Ｆモデルが必要となる。そこで、以下の手順により、Ｉ／Ｆモデルモデルを作成する。

【００７８】まず、図１３に示すようなメモリマップを作成する。このメモリマップは、SWにプログラムをコンパイルしたものである。HWは、レジスタをメモリにマッピングする。

【００７９】そして、各機能ブロックのアドレスを決定し、各機能ブロックのアドレステーブルをデータバース化して、メモリマップＤＢに登録する。そして、図１４に示すように、メモリマップＤＢに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する。

【００８０】バスコントローラは、各HWブロックのＰ（優先度）にしたがって、各バス毎のアービトレーション方法ごとに用意したＨＤＬテンプレートを生成する。すなわち、バスコントローラは、各ハードウェアのレジスタアドレスをメモリマップＤＢから抽出して、関数を作成し、これをドライバー関数テンプレートとしてSWモデルに格納する。

【００８１】図１５は、基本ハードウェアアクセス用のドライバー関数テンプレートの一例を示す図である。このようなメモリマップＤＢ，アドレスデコーダ，関数テンプレートを利用することにより、設計されたシステムのHWによってSWを動作させたり、システムのSWによってHWを動作させることが可能になり、HW／SW協調検証を行なうことができる。

【００８２】

【発明の効果】本発明により、ＩＰの機能に着目してより最適化された半導体集積回路装置を形成するための設計が可能になった。

【図面の簡単な説明】

【図１】（ａ），（ｂ），（ｃ）は、ＩＰ群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるＩＰ群の例を示す平面図、及びシリコン配線基板の断面図である。

20

＊【図２】本発明の実施形態におけるHW／SW協調設計の概略的な手順を示すフロー図である。

【図３】機能設計とＩＰ機能設計との詳細な処理の内容を説明するための部分フロー図である。

【図４】システムアーキテクチャ設計における処理の内容を説明するための部分フロー図である。

【図５】マッピングと性能解析の概略とを説明するための部分フロー図である。

【図６】各種のマッピングを行なった結果のコスト関数（Cost-tot）の値の例を示す図である。

【図７】静的なHW／SW分割を説明するための図である。

【図８】（ａ），（ｂ）は、動的なHW／SWの切り換えにおける消費電力Powerのタイムチャート図である。

【図９】消費電力を考慮したHW／SWの切り換え制御のためのプログラムの例をＣ言語で示す図である。

【図１０】（ａ），（ｂ）は、動的なHW／SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。

【図１１】処理速度を考慮したHW／SWの切り換え制御のためのプログラムの例をＣ言語で示す図である。

【図１２】（ａ），（ｂ）は、第２の実施形態の第５の具体例における機能の設定に関する半導体デバイスの構成を示す斜視図及びブロック回路図である。

【図１３】SWにプログラムをコンパイルして得られたメモリマップを示す図である。

【図１４】メモリマップＤＢに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する処理を示す図である。

【図１５】基本ハードウェアアクセス用のドライバー関数テンプレートの一例を示す図である。

【符号の説明】

１０　シリコン配線基板
１１　シリコン基板
１２　グランドプレーン
１３　第１配線層
１４　第２配線層
＊　１５　パッド

【図６】

| | MAP1 | MAP2 | MAP3 |
|---|---|---|---|
| Cost-tot | 80 | 100 | 103 |

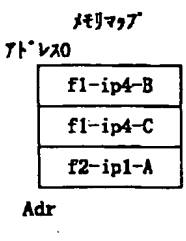
【図９】

```
if(Power(now)>MAXpower)
{
   regC=ON;
} else{
   regC=OFF;
   func_C( );
}
```

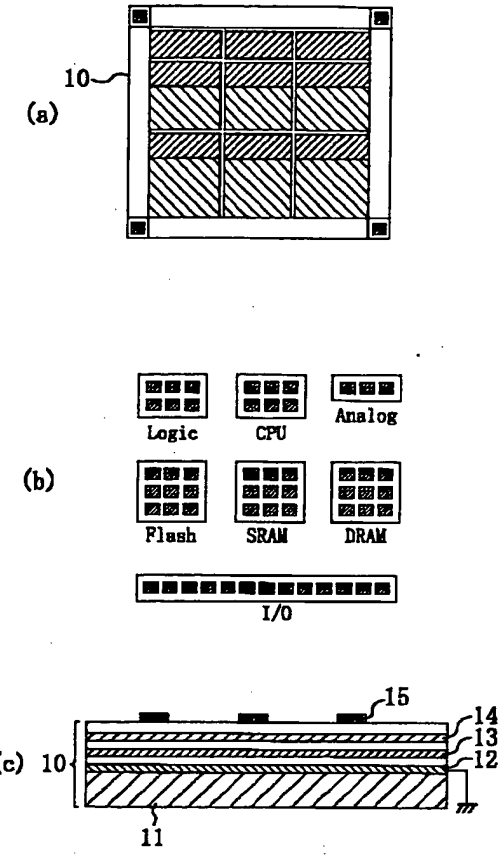
【図１１】

```
if(Load cpu>IDLE)
{
   regC=OFF;
   func_C( );
} else{
   regC=ON;
}
```

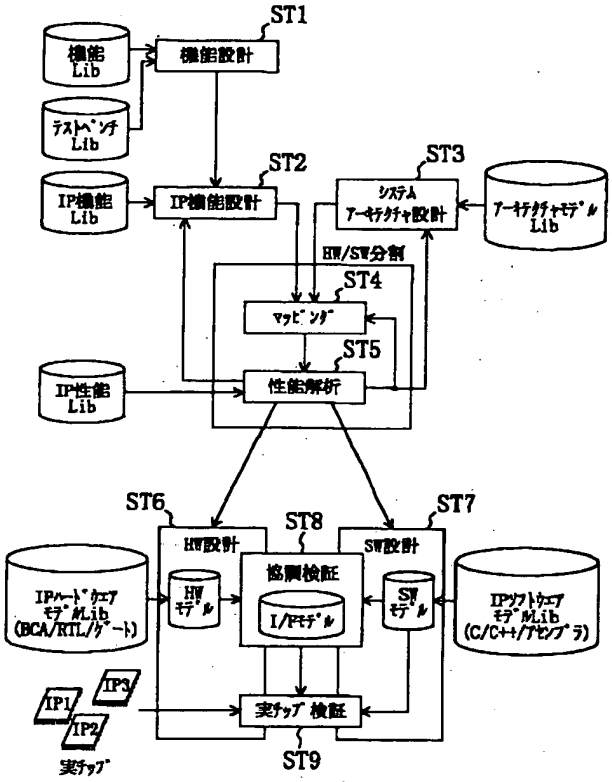
【図１３】



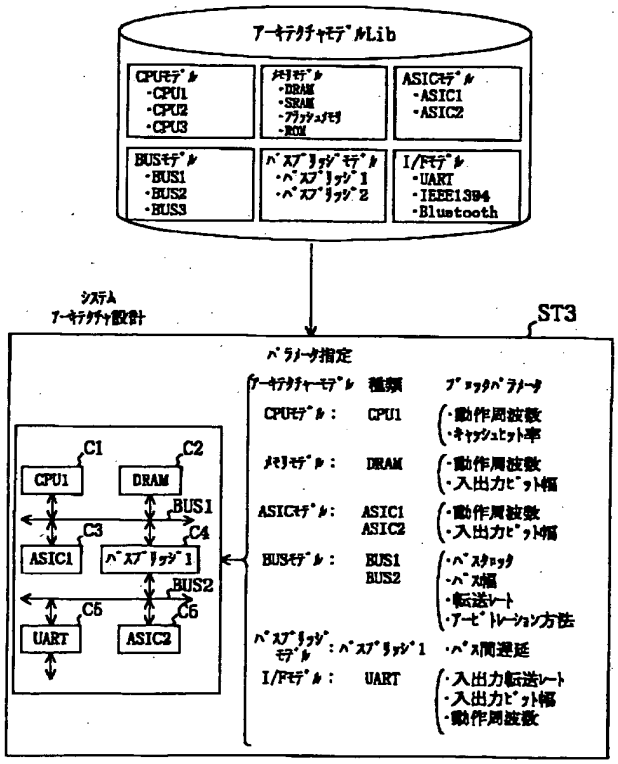
【図１】



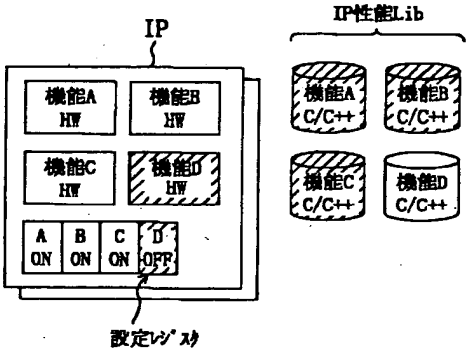

【図２】



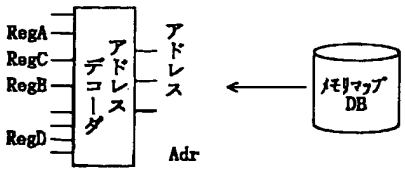

【図４】



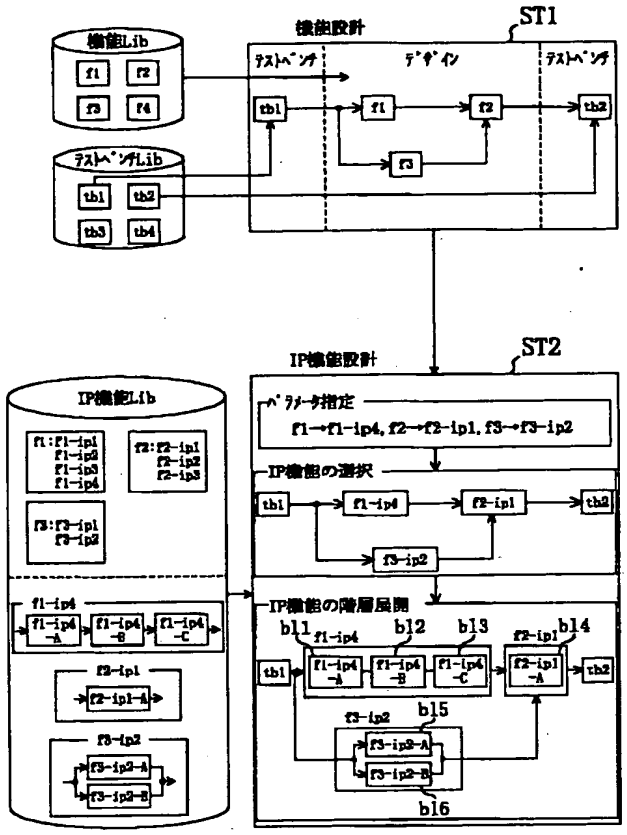

【図７】



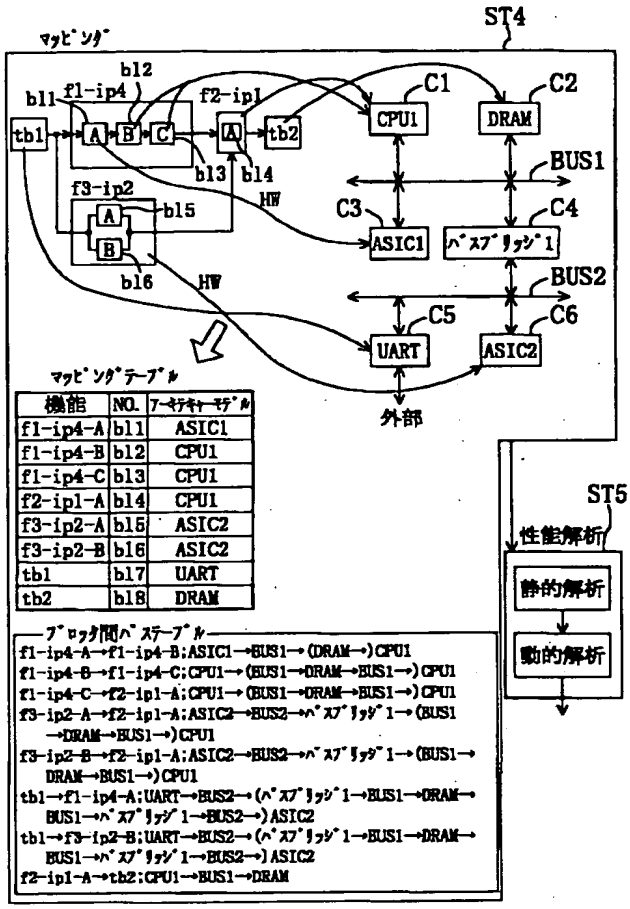

【図１４】



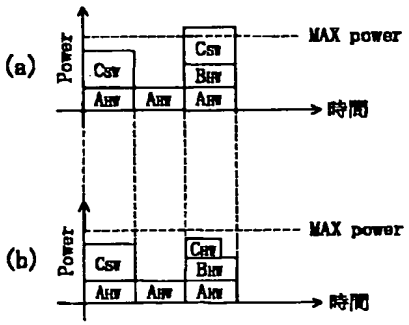
【図３】



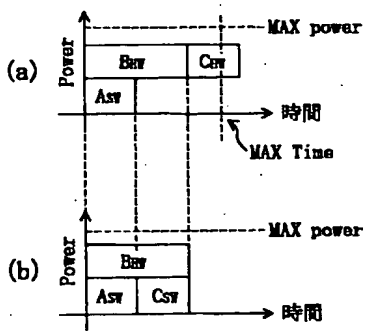

【図５】




【図８】



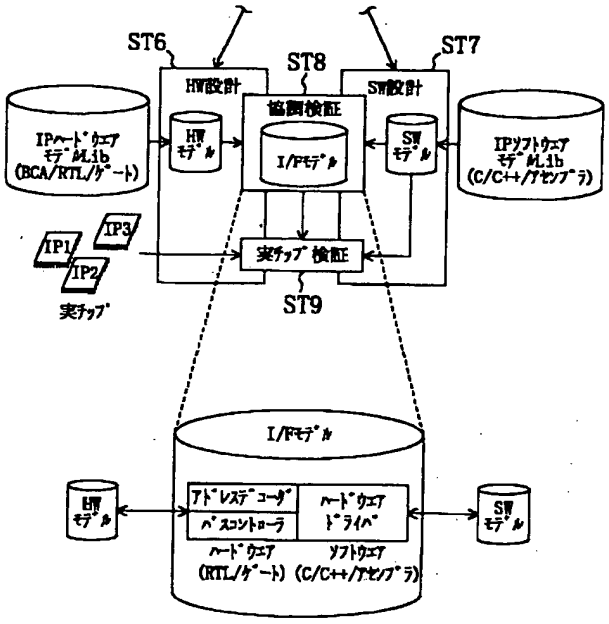

【図１０】




【図１５】

```
ドライバ関数テンプレート
read_mem8(  );write_mem8(  )
read_mem16(  );write_mem16(  )
```

【図１２】




【手続補正書】
【提出日】平成１３年１０月２日（２００１．１０．２）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】全文
【補正方法】変更
【補正内容】
【書類名】　　明細書
【発明の名称】　半導体集積回路装置の設計方法
【特許請求の範囲】
【請求項１】　共通の機能を有する複数のＩＰを、複数の機能について記憶装置に格納するステップ（ａ）と、
ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、
上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のＩＰから１つのＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）とを含む半導体集積回路装置の設計方法。
【請求項２】　請求項１記載の半導体集積回路装置の設計方法において、
上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のＨＷモデルとＳＷモデルとに割り当てるように分割するステップ（ｅ）とをさらに含むことを特徴とする半導体集積回路装置の設計方法。
【請求項３】　請求項２記載の半導体集積回路装置の設計方法において、
上記ステップ（ｃ）の後、上記ＩＰの階層を機能ブロックレベルに展開するステップ（ｄ）をさらに含み、
上記ステップ（ｅ）では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のＨＷモデルとＳＷモデルとに割り当てることを特徴とする半導体集積回路装置の設計方法。
【請求項４】　各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、
ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、
上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、
上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、
上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のＨＷ（ハードウェア）モデルとＳＷ（ソフトウェア）モデルとに割り当てるように分割するステップ（ｅ）とを含む半導体集積回路装置の設計方法。
【請求項５】　請求項４記載の半導体集積回路装置の設計方法において、
上記ステップ（ｅ）では、機能ブロックのＨＷの部分から面積の重複を考慮した静的解析を行なうことを特徴とする半導体集積回路装置の設計方法。

【請求項６】　請求項４記載の半導体集積回路装置の設計方法において、
上記ステップ（ｅ）の後、割り当てられた上記機能群構造中の各機能ブロックのＨＷモデルとＳＷモデルとへの割り当てを決定するための動的性能解析を行なうステップ（ｆ）をさらに含むことを特徴とする半導体集積回路装置の設計方法。
【請求項７】　請求項６記載の半導体集積回路装置の設計方法において、
上記データベースには、ＨＷの機能毎に当該ＨＷと等価な機能を有するＳＷモデルを格納しておくことを特徴とする半導体集積回路装置の設計方法。
【請求項８】　請求項７記載の半導体集積回路装置の設計方法において、
上記ステップ（ｆ）では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのＨＷモデルとＳＷモデルとへの割り当てを切り換えることを特徴とする半導体集積回路装置の設計方法。
【請求項９】　請求項６記載の半導体集積回路装置の設計方法において、
上記ステップ（ｆ）では、ある機能ブロックをＨＷに割り当てたときのＣＰＵの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをＳＷに切り換えることを特徴とする半導体集積回路装置の設計方法。
【請求項１０】　請求項５～９のうちいずれか１つに記載の半導体集積回路装置の設計方法において、
上記ステップ（ｅ）では、メモリの消費電力を考慮した解析を行なうことを特徴とする半導体集積回路装置の設計方法。
【請求項１１】　複数のＨＷのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、
上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）とを含み、
上記ステップ（ｂ）では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。
【請求項１２】　請求項１１記載の半導体集積回路装置の設計方法において、
上記ステップ（ｂ）では、上記バスブリッジモデルが、バス幅が互いに異なる２つのバス間をバス幅及びデータ転送速度を調整するように接続していることを特徴とする半導体集積回路装置の設計方法。
【請求項１３】　複数のＨＷのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、
上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）と、
上記仕様を満たすための機能群構造を構築するステップ（ｃ）と、
上記機能群構造の入力側及び出力側にテストベンチを付設するステップ（ｄ）と、
上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）とを含む半導体集積回路装置の設計方法。
【請求項１４】　請求項１３記載の半導体集積回路装置の設計方法において、
上記ステップ（ａ）では、上記複数のコンポーネントの中にＩ／Ｆモデルを含めておき、
上記ステップ（ｂ）では、上記コンポーネント群にＩ／Ｆモデルを含めるように上記アーキテクチャを構築することを特徴とする半導体集積回路装置の設計方法。
【請求項１５】　請求項１４記載の半導体集積回路装置の設計方法において、
上記ステップ（ｅ）では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のＩ／Ｆモデルに割り当てることを特徴とする半導体集積回路装置の設計方法。
【請求項１６】　請求項１３～１５のうちいずれか１つに記載の半導体集積回路装置の設計方法において、
上記ステップ（ａ）では、上記複数のコンポーネントの中にメモリ含めておき、
上記ステップ（ｂ）では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、
上記ステップ（ｅ）では、上記入力側又は出力側のテストベンチを上記アーキテクチャ中のメモリに割り当てることを特徴とする半導体集積回路装置の設計方法。
【請求項１７】　各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、
ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、
上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、
上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、
上記階層展開された各機能ブロックをアーキテクチャモデル中のコンポーネントに割り当てるステップ（ｅ）と、
上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャモデル中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ（ｆ）とを含む半導体集積回路装置の設計方法。
【請求項１８】　各々ある機能を有する複数のＩＰを記

憶装置に格納するステップ（ａ）と、
ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、
上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、
上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ（ｅ）と、
上記コンポーネントのバスアクセスのための優先度を設定するステップ（ｆ）と、
上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ（ｇ）とを含み、
上記静的な性能解析においては、上記各コンポーネントのアーキテクチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算することを特徴とする半導体集積回路装置の設計方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数の機能を有する回路を複数個配置した半導体集積回路装置の設計方法に関する。

【０００２】

【従来の技術】近年、複数のＬＳＩを共通の基板上に形成したシステムＬＳＩという概念が提起されており、システムＬＳＩの設計手法としても各種の提案がなされている。特に、システムＬＳＩの利点は、ＤＲＡＭなどのメモリや、ロジックＬＳＩや、高周波回路などのアナログ回路を１つの半導体装置内に収納して、多種，多機能の半導体装置を極めて高集積化して実現することができることである。

【０００３】

【発明が解決しようとする課題】ところで、上記従来のシステムＬＳＩにおいては、システムＬＳＩを設計する際に、ＩＰ，コアと呼ばれる設計資産を利用することが多い。そのとき、各コアは個別に設計されたものなので、これらを統合したときには必ずしも円滑な動作が得られないことがある。しかし、従来のコア又はＩＰは、その中身がブラックボックスとなっているので、円滑な動作を確保するシステムを設計するためには多大の手間と時間とを要しているのが現実である。

【０００４】本発明の目的は、ＩＰの機能の同一性を図りつつＩＰの選択を柔軟に行なうための手段を講ずることにより、最適化が容易な半導体集積回路装置の設計方法の提供を図ることにある。

【０００５】

【課題を解決するための手段】本発明の第１の半導体集積回路装置の設計方法は、共通の機能を有する複数のＩＰを、複数の機能について記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を共通に有する複数のＩＰから１つのＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）とを含んでいる。

【０００６】この方法により、機能が互いに共通する複数のＩＰをＬｉｂに格納しておいて、機能ごとに適当なＩＰを選択することができるので、ＩＰを代えても機能の同一性が保持される。したがって、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【０００７】上記各機能ごとに、当該機能の実現対象をデータベースのアーキテクチャモデル中のＨＷモデルとＳＷモデルとに割り当てるように分割するステップ（ｅ）とをさらに含むことにより、最適化された半導体集積回路装置を設計することができる。ができる。

【０００８】上記ステップ（ｃ）の後、上記ＩＰの階層を機能ブロックレベルに展開するステップ（ｄ）をさらに含み、上記ステップ（ｅ）では、上記展開された各機能ブロックごとに、当該機能ブロックの実現対象をアーキテクチャモデル中のＨＷモデルとＳＷモデルとに割り当てることにより、より最適化された半導体集積回路装置を設計することができる。

【０００９】本発明の第２の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、上記階層展開された各機能ブロックごとに、当該機能ブロックの実現化対象をデータベースのアーキテクチャモデル中のＨＷモデルとＳＷモデルとに割り当てるように分割するステップ（ｅ）とを含んでいる。

【００１０】このように、ＩＰを階層展開してからＨＷモデルとＳＷモデルとに分割することにより、ＩＰ内をＨＷモデルとＳＷモデルとに区別してマッピングを行なうことが可能となり、ＩＰ毎にマッピングする場合に比べて、より最適化されたシステムの設計が可能になる。

【００１１】上記ステップ（ｅ）では、機能ブロックのＨＷの部分から面積の重複を考慮した静的解析を行なうことにより、面積の計算精度の悪化を防止することができる。

【００１２】上記ステップ（ｅ）の後、割り当てられた上記機能群構造中の各機能ブロックのＨＷモデルとＳＷモデルとへの割り当てを切り換える動的性能解析を行なうステップ（ｆ）をさらに含むことにより、解析精度の向上を図ることができる。

【００１３】上記データベースには、HWの機能毎に当該HWと等価な機能を有するSWモデルを格納しておくことが好ましい。

【００１４】上記ステップ（ｆ）では、時間軸に沿って消費電力が常に上限値を超えないように各機能ブロックのHWモデルとSWモデルとへの割り当てを切り換えることができる。

【００１５】上記ステップ（ｆ）では、ある機能ブロックをHWモデルに割り当てたときのCPUの負荷率がある下限値よりも小さいときは、当該機能ブロックの割り当てをSWに切り換えることもできる。

【００１６】上記ステップ（ｅ）では、メモリの消費電力を考慮した解析を行なう。

【００１７】本発明の第３の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）とを含み、上記ステップ（ｂ）では、コンポーネント群に複数のバスと各バス間を接続するバスブリッジモデルとが含まれるように上記アーキテクチャを構築する方法である。

【００１８】このような複数のバスを設けることにより、IPOSデバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。

【００１９】上記ステップ（ｂ）では、上記バスブリッジモデルが、バス幅が互いに異なる２つのバス間をバス幅及びデータ転送速度を調整するように接続していることにより、当該バスブリッジモデルに接続される２つのバスのバス幅やデータ転送速度が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。

【００２０】本発明の第４の半導体装置集積回路装置の設計方法は、複数のHWのコンポーネントをアーキテクチャモデルとして記憶装置に格納するステップ（ａ）と、上記記憶装置からアーキテクチャモデルをコンポーネント群として取り出して、このコンポーネント群によりある仕様を満たすアーキテクチャを構築するステップ（ｂ）と、上記仕様を満たすための機能群構造を構築するステップ（ｃ）と、上記機能群構造の入力側及び出力側にテストベンチを付設するステップ（ｄ）と、上記機能群構造中の機能及びテストベンチを上記アーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）とを含んでいる。

【００２１】この方法により、テストベンチをHWにマッピングすることが可能になり、性能解析などにテストベンチの入出力に要する時間などのパラメータを含めることができる。

【００２２】上記ステップ（ａ）では、上記複数のコンポーネントの中にＩ／Ｆモデルを含めておき、上記ステップ（ｂ）では、上記コンポーネント群にＩ／Ｆモデルを含めるように上記アーキテクチャを構築することにより、入出力の信号を取り扱うＩ／Ｆモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【００２３】上記ステップ（ｅ）では、上記入力側のテストベンチを上記アーキテクチャ中のＩ／Ｆモデルに割り当てることにより、従来、単に入力信号として扱われていたテストベンチがHWであるＩ／Ｆモデルにマッピングされるので、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、Ｉ／Ｆが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【００２４】上記ステップ（ａ）では、上記複数のコンポーネントの中にメモリ含めておき、上記ステップ（ｂ）では、上記コンポーネント群にメモリを含めるように上記アーキテクチャを構築し、上記ステップ（ｅ）では、上記出力側のテストベンチを上記アーキテクチャモデル中のメモリに割り当てることもできる。

【００２５】本発明の第５の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記ＩＰを機能ブロックレベルに階層展開するステップ（ｄ）と、上記階層展開された各機能ブロックをアーキテクチャ中のコンポーネントに割り当てるステップ（ｅ）と、上記機能ブロックから当該機能ブロックの出力を受ける他の機能ブロックまでの経路を、上記アーキテクチャ中のコンポーネント間の経路として記述する機能ブロック間テーブルを作成するステップ（ｆ）とを含んでいる。

【００２６】このように機能ブロック間パステーブルを作成することにより、動的な性能解析を精度よく行なうことができる。

【００２７】本発明の第６の半導体装置集積回路装置の設計方法は、各々ある機能を有する複数のＩＰを記憶装置に格納するステップ（ａ）と、ある仕様を満たすための機能群構造を構築するステップ（ｂ）と、上記記憶装置から上記機能群構造中の各機能について、当該機能を有するＩＰを選択して取り出して、上記機能群構造中の各機能と置き換えるステップ（ｃ）と、上記機能群中の各機能をデータベースのアーキテクチャモデル中のコンポーネントに割り当てるステップ（ｅ）と、上記コンポーネントのバスアクセスのための優先度を設定するステップ（ｆ）と、上記機能群構造の各機能が上記コンポーネントに割り当てられた状態で上記機能群構造の静的な性能解析を行なうステップ（ｇ）とを含み、上記静的な性能解析においては、上記各コンポーネントのアーキテ

クチャモデル中におけるバスアクセスの優先度をトグル数と見なして上記機能群構造の消費電力を計算する方法である。

【００２８】優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００２９】

【発明の実施の形態】－ＩＰＯＳデバイスの基本的な構造－

ここで配線の断面積を大きく確保しつつ、多種，多機能のデバイスを内蔵した半導体装置を実現するための１つの手段として、配線層を有する半導体配線基板例えばシリコン配線基板（Ｓｕｐｅｒ－Ｓｕｂ）上に、各種デバイスを内蔵したチップＩＰを搭載する構成がある。その場合、各チップＩＰ内に設けられる回路（ＩＣ）は半導体装置の設計上ＩＰ(Intellectual Property) として扱うことができ、各種ＩＰを半導体配線基板上に貼り合わせたものと考えることができる。つまり、半導体デバイス全体は、“ＩＰ　Ｏｎ　Ｓｕｐｅｒ－Ｓｕｂ”であるので、シリコン配線基板とＩＰ群とを備えた半導体デバイス全体を“ＩＰＯＳデバイス”とする。

【００３０】図１（ａ），（ｂ），（ｃ）は、ＩＰ（チップＩＰ）群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるＩＰ群の例を示す平面図、及びシリコン配線基板の断面図である。図１（ａ），（ｂ）に示すように、シリコン配線基板１０上には各種ＩＰを搭載するための複数の領域が設けられており、、各領域には、例えば、Analog－ＩＰ，Logic －ＩＰ，ＣＰＵ－ＩＰ，Flash メモリ－ＩＰ，ＳＲＡＭ－ＩＰ，ＤＲＡＭ－ＩＰ，Ｉ／Ｏ－ＩＰなどの各種ＩＰ群がチップＩＰとして搭載可能となっている。図１（ｃ）に示すように、シリコン配線基板１０は、シリコン基板１１と、シリコン基板１０上に絶縁膜（図示せず）を挟んで設けられたグランドプレーン１２と、グランドプレーン１２の上に層間絶縁膜を挟んで設けられた第１配線層１３と、第１配線層１３の上に層間絶縁膜を挟んで設けられた第２配線層１４と、第２配線層１４の上にパッシベーション膜を挟んで設けられたパッド１５とを備えている。パッド１５，各配線層１３，１４及びグランドプレーン１２間は、それぞれコンタクト（図示せず）を介して所望の部位で互いに接続されている。そして、各ＩＰは、パッド１５上に貼り付けられて、各ＩＰが配線層１３，１４により互いにあるいはグランドプレーン１２に電気的に接続される構造となっている。

【００３１】シリコン配線基板１０内の配線層１３，１４の寸法の制約は緩やかであり、数μｍ幅の配線をも設けることができるので、以下のような効果がある。経験的に、今までの半導体集積回路装置の微細化が進展した過程において、もっとも配線としての特性が良好であった世代の寸法を有する配線を設けることが可能となる。また、配線の電気インピーダンスを低減することができる。

【００３２】そして、シリコン配線基板上のチップＩＰは多くの機能を有しているが、これに対しては２つの考え方がある。１つは、できるだけ多くの機能を使用するという考え方であり、もう１つは、使用する機能を制限するという考え方である。つまり、ユーザの多様な要求に応えるためには、多種の使用方法に対応できる構成を有していることが好ましいが、反面、実際の使用に際してはいずれかの機能に限定する必要がある。つまり、この２つの相反する要求を満足させることにより、ユーザの多様な要求に応えつつ、少品種，大量生産に適したＩＰＯＳデバイスを構築することができるのである。

【００３３】したがって、後に選択，制限，切り換え，設定などの処理を柔軟に行なうことを前提として、つまり、機能に関する処理の柔軟性を保持しつつ，多種の機能をハードウェア（ＨＷ），ソフトウェア（ＳＷ）に分けて各ＩＰ（チップＩＰ）に組み込むことが重要となる。

【００３４】そこで、以下の実施形態においては、ＩＰＯＳデバイスを例とした大規模なシステム（以下、「ＩＰＯＳシステム」という）に適したＨＷ／ＳＷ協調設計について説明する。ただし、本発明は、必ずしもＩＰＯＳシステムだけでなく、従来のいわゆるシステムＬＳＩなどの大規模な集積回路システムの設計にも適用が可能である。

【００３５】（実施形態）

－ＨＷ／ＳＷ協調設計全体の概略－

図２は、本発明の実施形態におけるＨＷ／ＳＷ協調設計の概略的な手順を示すフロー図である。

【００３６】ステップＳＴ１で、機能ライブラリ（機能Ｌｉｂ）やテストベンチを用いて機能設計を行ない、ステップＳＴ２で、ＩＰ機能Ｌｉｂを用いたＩＰ機能設計を行なう。このステップＳＴ１，ＳＴ２では、半導体デバイスに搭載される各ＩＰに要求される機能が例えばＣ言語により記述される。例えばＪＰＥＧの場合には、信号処理（アルゴリズム）だけが記載されているような状態である。

【００３７】次に、ステップＳＴ３で、アーキテクチャモデルＬｉｂを用いてシステムのアーキテクチャ設計を行なう。システムのアーキテクチャモデルＬｉｂには、アーキテクチャモデルがシステムの設計に必要なＨＷ，ＳＷモデルとして存在しているので、このステップＳＴ３では、各モデルをバスに接続してシステム全体の概略的な構造を作り上げる。

【００３８】次に、ステップＳＴ４で、各機能（機能ブロック）をアーキテクチャのコンポーネントに割り当てるマッピングを行なった後、ステップＳＴ５で、性能Ｌｉｂを用いて性能解析を行ない、その結果、不具合があ

ればステップＳＴ３のアーキテクチャ設計，又はステップＳＴ４のマッピングに戻り、性能が満足されるまでこれらのステップを繰り返すループ処理を行なう。

【００３９】次に、ステップＳＴ６，ＳＴ７で、それぞれＩＰハードウェアモデルＬｉｂを用いたＨＷ設計と、例えばＩＰソフトウェアモデルＬｉｂを用いたＳＷ設計とを個別に行なってから、ＨＷ及びＳＷが詳細化されてくると、ステップＳＴ８で、Ｉ／Ｆモデルを用いたＨＷ／ＳＷ協調検証を行なう。その後、ステップＳＴ９で、実チップを用いて実チップの機能を確認するための実チップ検証を行なう。

【００４０】－機能設計及びＩＰ機能設計－

図３は、機能設計を行なうステップＳＴ１と、ＩＰ機能設計を行なうステップＳＴ２との詳細な処理の内容を説明するための部分フロー図である。

【００４１】ステップＳＴ１においては、以下の手順により、機能設計を行なう。機能Ｌｉｂには、ｆ１，ｆ２，ｆ３，ｆ４などの種々の機能（ｆｕｎｃｔｉｏｎ）が格納されている。この機能ｆ１，ｆ２，ｆ３，ｆ４とは、例えば画像処理におけるＭＰＥＧＫＬ，ＪＰＥＧ，Blue-toothなどの機能である。また、テストベンチＬｉｂには、ｔｂ１，ｔｂ２，ｔｂ３，ｔｂ４などの種々のテストベンチが格納されている。このテストベンチｔｂ１，ｔｂ２，ｔｂ３，ｔｂ４は、機能設計が適正か否かを判断するための与えられるデータである。例えば、図３のステップＳＴ１中に示すように、機能ｆ１，ｆ２，ｆ３を組み合わせたデザインがある場合には、入力テストベンチとして、テストベンチからｔｂ１を取り出して、これを入力データとして与える。また、出力データとして例えば画像出力のテストベンチｔｂ２を与える。その結果、ここで作り上げたデザインが適正かどうかがある程度検証される。

【００４２】次に、ステップＳＴ２におけるＩＰ機能設計は以下の手順で行なわれる。ＩＰ機能Ｌｉｂには、各機能ｆ１，ｆ２，ｆ３，ｆ４ごとに、この機能を実現するためのいくつかのＩＰが格納されている。例えば、機能ｆ１に対してｆ１－ｉｐ１～ｆ１－ｉｐ４があり、機能ｆ２に対してｆ２－ｉｐ１～ｆ２－ｉｐ３があり、機能ｆ３に対してｆ３－ｉｐ１，ｆ３－ｉｐ２がある。そこで、おおよその見当をつけて、デザイン中の機能ｆ１，ｆ２，ｆ３に対して用いるＩＰを指定する。例えば、機能ｆ１に対してはｆ１－ｉｐ４を、機能ｆ２に対してはｆ２－ｉｐ１を、機能ｆ３に対してはｆ３－ｉｐ２を用いるというようにパラメータを指定する。

【００４３】次に、ステップＳＴ１で作り上げられたデザイン中の各機能ｆ１，ｆ２，ｆ３に、それぞれｆ１－ｉｐ４，ｆ２－ｉｐ１，ｆ３－ｉｐ２を当てはめる。つまり、ＩＰ機能の選択を行なう。この段階では、選び出された各ＩＰはいずれもステップＳＴ１における機能ｆ１，ｆ２，ｆ３を有しているので、ステップＳＴ１で検証した結果と同じ結果が得られる。

【００４４】このように、機能が互いに共通する複数のＩＰをＬｉｂに格納しておいて、機能ごとに適当なＩＰを選択することにより、ＩＰを代えても機能の同一性が保持されるので、従来大きかった設計変更の確率をできるだけ小さくすることができる。

【００４５】次に、ＩＰ機能をさらに詳細に個々の機能に分けるＩＰ機能の階層展開を行なう。例えば、ｆ１－ｉｐ４について階層展開をすると、ｆ１－ｉｐ４－Ａ（例えば逆コサイン変換）を行なってからｆ１－ｉｐ４－Ｂ（例えば量子化）を行ない、その後ｆ１－ｉｐ４－Ｃ（例えばハフマン符号化）を行なうという機能になっていたとする。つまり、ＩＰであるｆ１－ｉｐ４を機能ブロックｆ１－ｉｐ４－Ａ，ｆ１－ｉｐ４－Ｂ，ｆ１－ｉｐ４－Ｃに展開するのである。同様に、ｆ２－ｉｐ１，ｆ３－ｉｐ２についても階層展開を行なって、ｆ２－ｉｐ１についてはｆ２－ｉｐ１－Ａという機能ブロックが、ｆ３－ｉｐ２についてはｆ３－ｉｐ２－Ａ，ｆ３－ｉｐ２－Ｂという機能ブロックがそれぞれ得られたとする。そして、この階層展開された各機能ブロックをデザイン中の各機能ｆ１，ｆ２，ｆ３に置換する。この階層展開されたＩＰの中身をＨＷ／ＳＷ分割するのが本発明の１つの特徴である。

【００４６】－アーキテクチャ設計－

次に、図４は、ステップＳＴ３における処理の内容を説明するための部分フロー図である。同図に示すように、アーキテクチャモデルＬｉｂには、ＣＰＵ，メモリ，ＡＳＩＣ（ロジック回路），ＢＵＳ，バスブリッジモデル，Ｉ／Ｆモデルなどのコンポーネント（ｃ）についての多くの種類が格納されている。そこで、このコンポーネントごとに各種のコンポーネントパラメータをもっているので、コンポーネントパラメータの値を与える。つまり、ＣＰＵについては動作周波数，キャッシュヒット率があり、ＤＲＡＭについては動作周波数，ビット幅があり、ＡＳＩＣについては動作周波数，ビット幅があり、ＢＵＳについてはバスクロック，バス幅，転送レート（ｂｐｓ），アービトレーション方法があり、バスブリッジモデルについてはバスブリッジ間遅延があり、Ｉ／ＦモデルとしてＵＡＲＴを選択したときには入出力転送レート（ｂｐｓ），入出力ビット幅，動作周波数，ビット幅がある。そして、各コンポーネントパラメータに具体的な値を入れる。そして、このパラメータ指定を行なってから、例えば図４に示すように、ＣＰＵ１（ｃ１），ＤＲＡＭ１（ｃ２），ＡＳＩＣ１（ｃ３），ＡＳＩＣ２（ｃ６）及びＵＡＲＴ（ｃ５）をＢＵＳ１，ＢＵＳ２及びバスブリッジ１（ｃ４）を介して接続した構造を有するアーキテクチャを作り上げる。

【００４７】ここで、このアーキテクチャの特徴は、複数のバス（ＢＵＳ１，ＢＵＳ２）を備え、ＢＵＳ１－ＢＵＳ２間を接続するバスブリッジ１（ｃ４）をコンポー

ネントとして設けている点である。バスは３つ以上あってもよく、このような複数のバスを設けることにより、半導体デバイスのような半導体配線基板を有する構造に適したアーキテクチャを構築することができる。複数のバスの例としては、標準バスに低速バス，高速バスなどを別途設けたものが考えられる。そして、バスブリッジモデルを設けることにより、当該バスブリッジモデルに接続される２つのバスのビット幅が互いに異なる場合などにおいてもデータの円滑な転送を図ることができる。バスブリッジモデルの例としては、１６ビット幅のＢＵＳ（例えばバス１）のデータを８ビットずつ別のＢＵＳ（例えばＢＵＳ２）に送り込むバッファがある。また、アーキテクチャにおいて、Ｉ／ＦモデルをＡＳＩＣとは別に設けたことにより、入出力の信号を取り扱うＩ／Ｆモデルにおける信号の遅延や消費電力を考慮したシステムの最適化が可能になる。

【００４８】－マッピング－

次に、図５は、ステップＳＴ４，ＳＴ５におけるマッピングと性能解析の概略とを説明するための部分フロー図である。

【００４９】同図に示すように、ステップＳＴ４においては、ステップＳＴ２のＩＰ機能設計によって得られたデザイン中の階層展開されたＩＰを、ＨＷとＳＷとに割り当てる処理であるマッピングを行なう。例えば、テストベンチｔｂ１は、ＨＷとしてＩ／ＦモデルであるＵＡＲＴに割り当てられる。また、テストベンチｔｂ２は、ＨＷとしてＤＲＡＭ１に割り当てられる。また、階層展開されているｆ１－ｉｐ４中の機能ブロックｆ１－ｉｐ４－Ａ（ｂｌ１）はＨＷとしてＡＳＩＣ１に、機能ブロックｆ１－ｉｐ４－Ｂ（ｂｌ２），機能ブロックｆ１－ｉｐ４－Ｃ（ｂｌ３）はＳＷとしてＣＰＵ１に、ｆ２－ｉｐ１中の機能ブロックｆ２－ｉｐ１－Ａ（ｂｌ４）はＳＷとしてＣＰＵ１に、ｆ２－ｉｐ２中の機能ブロックｆ３－ｉｐ２－Ａ（ｂｌ５），機能ブロックｆ２－ｉｐ２－Ａ（ｂｌ６）はすべてＨＷとしてＡＳＩＣ２にそれぞれ割り当てられる。

【００５０】ここで、このマッピングの特徴は、テストベンチをＩ／Ｆモデルにマッピングしたことである。従来、単に入力信号として扱われていたテストベンチをＨＷであるＩ／Ｆモデルにマッピングすることにより、後の性能解析の際に遅延を考慮した処理時間の解析が可能になり、また、Ｉ／Ｆモデルが動作することを考慮した消費電力の解析が可能になる。つまり、解析精度の向上を図ることができる。

【００５１】また、このように、１つのＩＰをＨＷとＳＷとにわけてマッピングができるようになったのは、ＩＰ機能設計において、上述のようにＩＰを機能ブロックのレベルまで階層展開したからである。そして、マッピングに応じて、図５の左下に示すように、各機能とデザイン中のコンポーネントとのマッピング対応表が機能番号（ａ１～ａｎ）とともに作成される。また、例えば機能ブロックｆ１－ｉｐ４－Ａ→機能ブロックｆ１－ｉｐ４－Ｂというパスは、ＡＳＩＣ１→ＢＵＳ１→（ＤＲＡＭ１→）ＣＰＵ１というデータのパスとなり、機能ブロックｆ１－ｉｐ４－Ｂ→機能ブロックｆ１－ｉｐ４－Ｃというパスは、ＣＰＵ１→（ＤＲＡＭ１→）ＣＰＵ２というデータのパスとなることが、ブロック間パステーブルとして作成され、設計装置の記憶装置（図示せず）に格納される。なお、括弧内のコンポーネントはその部分が経由される場合もあり、経由されない場合もあることを示している。

【００５２】ここで、このように機能ブロック間パステーブルを作成することにより、後述するような動的な性能解析を精度よく行なうことができる。

【００５３】次に、このようなマッピング結果を用いてステップＳＴ５における性能解析を行なう。このステップにおいては、静的解析と動的解析とを行なってもよいし、静的解析及び動的解析のうちいずれか一方のみを行なってもよい。

【００５４】－性能解析（静的解析）－

性能解析のうち静的解析は、例えば所望のパラメータを変数としてシステム全体のコスト関数（Cost-tot）を定義して、このコスト関数（Cost-tot）を求める。そして、コスト関数（Cost－tot）が最小又はある上限値以下になるかを目標として設計を進める。このコスト関数（Cost-tot）の一例として、例えば下記式（１）

$$\begin{aligned} Cost\text{-}tot = {} & k1\ (Speed_{HW\text{-}tot} + Speed_{SW\text{-}tot}) \\ & k2\ (Area_{HW\text{-}tot} + Area_{SW\text{-}tot}) \\ & k3\ (Power_{HW\text{-}tot} + Power_{SW\text{-}tot}) \end{aligned} \qquad (1)$$

がある。ここで、ｋ１，ｋ２，ｋ３は重み係数であり、$Speed_{HW\text{-}tot}$はＨＷ全体の処理速度であり、$Speed_{SW\text{-}tot}$はＳＷ全体の処理速度であり、$Area_{HW\text{-}tot}$はＨＷ全体の面積であり、$Area_{SW\text{-}tot}$はＳＷ全体の面積であり、$Power_{HW\text{-}tot}$はＨＷ全体の消費電力であり、$Power_{SW\text{-}tot}$はＳＷ全体の消費電力である。つまり、この例では、パラメータである処理速度ができるだけ大きく，面積ができるだけ小さく、消費電力ができるだけ小さくなるようにシステムを構築したいと考えていることになる。そこで、コスト関数（Cost-tot）は小さいほどよいとする。例えば、各種のマッピングＭＡＰ１，ＭＡＰ２，ＭＡＰ３を行なった結果、コスト関数（Cost-tot）が図６に示す値になったとする。このときには、もっとも値が小さいマッピングＭＡＰ１が最適であると判断する。

【００５５】ここで、例えば処理速度に関する計算は、マッピング結果を用いて、下記式（２）

$$Speed_{HW\text{-}tot}(f_{b11}\sim f_{b16},\ f_{bus},\ R_{bus},\ P_{b11}\sim P_{b16})$$
$$=Speed_{HW\text{-}b11}(f_{c6},\ f_{bus},\ R_{bus},\ R_{c6})+$$
$$\cdots\cdots+Speed_{HW\text{-}b16}(f_{c6},\ f_{bus},\ R_{bus},\ P_{c6}) \quad (2)$$

のように行なわれる。ここで、式（２）において、図５に示すように、マッピングしたときの機能ブロック番号をｂｌ１～ｂｌ６とし、コンポーネント番号をｃ１～ｃ６とする。そして、ｆは動作周波数であり、Ｒはバスの転送レートであり、Ｐは機能のパスに対する優先順位（優先度）（０～１の値）を表している。この優先順位Ｐは、パスに接続されるるすべてのコンポーネントの優先度を合計すると１になるように設定されている。

【００５６】すると、式（２）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の処理速度は、下記式（３）

$$Speed_{HW\text{-}b11}(f_{c6},\ f_{bus},\ R_{bus},\ P_{c6})$$
$$=1/(f_{c6}\times f_{bus}\times R_{bus}\times P_{c6}) \quad (3)$$

のように求められる。つまり、処理速度はできる限り小さい方が好ましいことから、動作周波数ｆ，転送レートＲ，優先順位Ｐの逆数からコスト関数（Cost-tot）のうち処理速度Ｓpeedの成分を計算する。

【００５７】次に、式（１）中のＳＷについての処理速度成分$Speed_{SW\text{-}tot}$は、マッピングのブロック間パステーブルに基づいて、下記式（４）

$$Speed_{SW\text{-}tot}(St_{CPU\text{-}c1}\sim St_{CPU\text{-}c6},\ f_{CPU})$$
$$=Speed_{SW\text{-}b11}(St_{CPU\text{-}c1},\ f_{CPU})+$$
$$\cdots+Speed_{SW\text{-}b16}(St_{CPU\text{-}c6},\ f_{CPU}) \quad (4)$$

により求められる。この計算の考え方は以下の通りである。機能ブロックをｆｕｎｃｔｉｏｎによって表すと、処理はＣ言語によって記述することができ、Ｃ言語で記述すればアセンブリ言語にコンパイルすることが可能となる。したがって、アセンブラのステップ数で処理速度を演算することができるからである。

【００５８】ここで、すると、式（４）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の処理速度は、下記式（５）

$$Speed_{SW\text{-}b11}(St_{CPU\text{-}c1},\ f_{CPU})=\underline{St_{CPU\text{-}c1}}/f_{CPU} \quad (5)$$

のように求められる。このステップ数が小さいほど速度が大きいので、式（５）の数値がなるべく小さいことが望ましいことになる。

【００５９】次に、面積に関する計算は、マッピング結果を用いて、下記式（６）

$$Area_{HW\text{-}tot}=Area_{HW\text{-}f1\text{-}ip4}$$
$$+Area_{HW\text{-}f2\text{-}ip1}+Area_{HW\text{-}f3\text{-}ip2} \quad (6)$$

のように行なわれる。つまり、ＩＰ機能を選択した後でＩＰ機能の階層展開を行なう前に使用される各ＩＰ（ｆ１－ｉｐ４，ｆ２－ｉｐ１，ｆ３－ｉｐ２）の情報があるので、これからHW部分の面積を求めることができる。

【００６０】次に、機能ブロックをコンパイルすると、アセンブラでメモリサイズがわかるので、ＳＷのメモリの部分をすべて加算しておいて、それにエリアＤＲＡＭというデータベースを用いてＤＲＡＭの１ビット当たりの面積（単価）を計算する。それに、メモリサイズを乗じると、下記式（７）

$$Area_{SW\text{-}tot}=(Msize_{SW\text{-}b11}+\cdots+Msize_{SW\text{-}b16})\times Area_{DRAM} \quad (7)$$

により、求めることができる。

【００６１】このように、ＩＰ内が機能ブロックに分割されたことにより、機能ブロックのＳＷの部分からメモリサイズを計算すると、各機能ブロック間で面積を重複して計算する場合があるので、これを考慮して面積の計算精度の悪化を防止するのである。

【００６２】次に、HW部分の消費電力に関する計算は、マッピング結果を用いて、下記式（８）

$$Power_{HW\text{-}tot}=Power_{HW\text{-}b1}+\cdots+Power_{HW\text{-}b16} \quad (8)$$

のように行なわれる。すると、式（８）中の例えば機能ブロックｂｌ１（ｆ１－ｉｐ４－Ａ）の消費電力は、下記式（９）

$$Power_{HW\text{-}b11}(f_{c6},\ V_{c6})$$
$$=\alpha_{b11}\times f_{c6}\times(V_{c6})^2\times P_{c6} \quad (9)$$

により、表される。ただし、αは機能ブロックｂｌ１のパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$f_{c6}$はコンポーネントｃ３（ＡＳＩＣ１）の動作周波数、$V_{c6}$はコンポーネントｃ３（ＡＳＩＣ１）の電圧、$P_{c6}$はコンポーネントｃ３（ＡＳＩＣ１）の優先順位（優先度）を示している。この優先度をトグル数の代わりに用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００６３】また、ＳＷ部分の消費電力に関する計算は、下記式（１０）

$$Power_{SW\text{-}tot} = Power_{SW\text{-}CPU\text{-}tot} + Power_{SW\text{-}MEM\text{-}tot} \quad (10)$$

により求められる。式（１０）の第１項は、

$$Power_{SW\text{-}CPU\text{-}tot} = \alpha_{CPU} \times f_{CPU} \times Load_{CPU} \quad (11)$$

となる。ここで、$\alpha_{CPU}$はＣＰＵのパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$Load_{CPU}$はＣＰＵロード（０～１）であって、「ＳＷにマッピングされた機能ブロックのメモリサイズの合計（ｂｉｔ）」を「すべての機能ブロックをＳＷにマッピングしたときのメモリサイズの合計（ｂｉｔ）」で除したものである。

【００６４】一方、式（１０）の第２項は、

$$Power_{SW\text{-}MEM\text{-}tot} = \alpha_{DRAM} \times f_{DRAM} \times (V_{DRAM})^{2} \times P_{DRAM} \quad (12)$$

となる。$\alpha_{DRAM}$はＤＲＡＭのパワー係数でＩＰ性能Ｌｉｂから抽出されるもの、$f_{DRAM}$はＤＲＡＭの動作周波数（Ｈｚ）、$V_{DRAM}$はＤＲＡＭの電圧、$P_{DRAM}$はＤＲＡＭの優先順位を示している。この優先度を用いることで、トグル数がわからない静的解析においても、ある程度の確度で消費電力が求まることになる。

【００６５】そして、以上の計算を行なって上で、式（１）から最終的に決定されるコスト関数（Cost-tot）が最小になるか、下限値以下になったときに静的解析に関する限り、設計されたシステムが適切であると判断する。

【００６６】－性能解析（動的解析）－

動的解析においては、面積Ａｒｅａについては静的解析と同じであるので、処理速度Ｓｐｅｅｄと消費電力Ｐｏｗｅｒのみを求める。動的解析においては、設計システムに実際にデータを入力させて動作させるシミュレーションを行なう。したがって、実行時間（Ｔｉｍｅ（ｓｅｃ））や、各機能ブロックの入出力データ、トグル率がシミュレーションから得られる。

【００６７】そこで、動的解析においては、基本的には、この得られた要素を上記式（１）～（１２）に代入することにより、コスト関数（Cost-tot）の値を求めることができる。その際、式（１）における（$Speed_{HW\text{-}tot}$＋$Speed_{SW\text{-}tot}$）は、シミュレーションで得られた時間Ｔｉｍｅに置き換えることができ、式（９）における各コンポーネントの優先順位Ｐはシミュレーションで得られた各コンポーネントのトグル数で置き換えることができる。そして、式（１１）におけるＣＰＵロード$Load_{CPU}$は、実際のシミュレーションから求まる平均ロードを用いる。

【００６８】さらに、式（１２）におけるＤＲＡＭの優先順位Ｐ（優先度）はシミュレーションで得られたＤＲＡＭのトグル数で置き換えることができる。

【００６９】－ＨＷ／ＳＷ分割（静的分割）－

次に、ステップＳＴ４，ＳＴ５と共に行なわれるＩＰ内のＨＷ／ＳＷ分割について説明する。ＨＷ／ＳＷには静的分割と動的分割とがあるが、まず、静的な分割について説明する。

【００７０】図７は、静的なＨＷ／ＳＷ分割を説明するための図である。同図に示すように、半導体デバイスなどのシステムに利用されるハードＩＰには、機能Ａ，機能Ｂ，機能Ｃ，機能Ｄがあり、かつ、これらの機能のＯＮ・ＯＦＦを設定するための設定レジスタがある。ただし、ＨＷ／ＳＷ分割における「機能」とは本実施形態における「機能ブロック」を意味している。一方、ＩＰ性能Ｌｉｂには、各機能Ａ，機能Ｂ，機能Ｃ，機能Ｄについてのモデルが例えばＣ／Ｃ＋＋言語によって記述されて格納いる。このとき、各機能をＨＷとＳＷとによって動作させた結果、１つの機能を変更，修正，削除する必要が生じる場合がある。ここで、例えば、設定レジスタ内の機能ＤをＯＦＦにすることで、例えば機能Ｄへのクロックの供給を停止するなどの制御を行なって、機能Ｄが動作しないようにすることができる。これにより、無駄な電力の消費を抑制して消費電力の低減を図ることができる。また、各機能Ａ～Ｄ間の接続を変更することもできる。

【００７１】さらに、例えば１つの機能ＤをＨＷからＩＰ性能Ｌｉｂに用意されたＳＷモデルに切り換えたり、用意しているＬｉｂ内のＳＷモデルを修正することもできる。つまり、いずれかのＬｉｂ内に各機能と等価なＳＷモデルを準備しておくことにより、１つのＩＰ内をＨＷ／ＳＷ分割することができ、ハードＩＰの機能が不足したときにも設計を進めることができる。また、ＳＷの機能を修正，追加することにより、システムの変更に柔軟に対応することができる。

【００７２】－動的なＨＷ／ＳＷ分割－

次に、動的なＨＷ／ＳＷ分割について説明する。図７に示すＨＷ／ＳＷ分割は、ＨＷ，ＳＷの設定を行なった後は、原則としてその後の修正は行なわない。それに対し、動的なＨＷ／ＳＷ分割，つまりＨＷ／ＳＷ動作の切り換えにおいては、機能（機能ブロック）が動作しながら柔軟に変更，追加，修正を受けることができる。このような動的なＨＷ／ＳＷ分割の例として、以下、第１～第３の具体例について説明する。

【００７３】１．第１の具体例

第１の具体例では、図７に示すＩＰにおいて、各機能のＯＮ，ＯＦＦをＳＷによって制御し、かつ、停止させる場合はＩＰ性能Ｌｉｂから同等のＳＷモデルを呼びだしてこれを用いる。設定レジスタへの制御信号により、各機能を動作させながらＨＷ／ＳＷの切り換えを行なうことができる。

【００７４】２．第２の具体例

図８（ａ），（ｂ）は、動的なＨＷ／ＳＷの切り換えにおける消費電力Ｐｏｗｅｒのタイムチャート図である。ま

た、図９は、このHW／SWの切り換え制御のためのプログラムの例をC言語で示す図である。図９に示すように、設定レジスタの制御により、ハードＩＰの機能Ｃを動作させた場合に、図８（ａ）に示すように、ＩＰの消費電力Powerが消費電力の上限値ＭＡＸpower を越える時があるとする。動的解析からどの機能が動作しているかが時間軸に応じてわかるので、あるレジスタに各機能の消費電力の合計を計算しながら書き込んでおいて、その値が上限値ＭＡＸpower を超えるときを検出すればよい。そして、ＩＰの消費電力Powerが消費電力の上限値ＭＡＸpower を越えると、図９に示すように、設定レジスタの制御によりハードＩＰの機能ＣをＯＦＦにして、代わりにSWモデルからｆｕｎｃ－Ｃを呼び出してこちらを動作させる。これにより、図８（ｂ）に示すように、消費電力Powerが上限値ＭＡＸpower を越えることがなくなり、所望の制約条件の下での作動を確保することができる。つまり、この例では、当該ＩＰの機能Ａ，Ｃが動作する期間には機能ＣがSWによって実現にされるが、機能Ａ，Ｂ，Ｃが動作する期間には機能ＣがHWによって実現されるうに設計することになる。

【００７５】３．第３の具体例

図１０（ａ），（ｂ）は、動的なHW／SWの切り換えにおける処理速度Speedの消費電力依存性を示す図である。また、図１１は、このHW／SWの切り換え制御のためのプログラムの例をC言語で示す図である。図１１に示すように、ＣＰＵの設定レジスタの制御により、ＣＰＵのロード（％）がある値ＩＤＬＥよりも大きいときには機能ＣをSWにより実行し、ＣＰＵのロード（％）がある値ＩＤＬＥ以下のときに機能ＣをHWにより実行するよう制御する。つまり、機能Ｃを常にHWで実行すると、図１０（ａ）に示すように処理時間Time が上限値ＭＡＸtimeを越える事態が発生する。そこで、レジスタにＣＰＵのロード（％）を検知しながら書き込んでおいて、ＣＰＵのロード（％）が下限値ＩＤＬＥを越えるときを検出しながら、ＣＰＵのロード（％）がある値ＩＤＬＥ以下のときには機能ＣをHWにより実行する。そして、ＣＰＵのロード（％）が下限値ＩＤＬＥを越えると、設定レジスタの制御によりハードＩＰの機能ＣをＯＦＦにして、代わりにSWモデルからｆｕｎｃ－Ｃを呼び出してSWを動作させる。このような制御を行なうことにより、図１０（ｂ）に示すように、処理時間Time が上限値ＭＡＸtimeを越えることがなくなり、所望の制約条件の下での作動を確保することができる。

【００７６】－HW／SW協調検証－

次に、ステップＳＴ８の協調検証について説明する。上述のステップＳＴ４のマッピング処理，ステップＳＴ５の性能解析処理に基づいて最終的に決定されたHW，SWの構成に従い、ステップＳＴ６，ＳＴ７で、HW設計とSW設計とを行なって、HWモデルとSWモデルとをそれぞれ作成する。HWモデルを作成する際にはＩＰハードウェアモデルＬｉｂ（ＢＣＡ，ＲＴＬ又はゲートレベルで記述）を利用し、SWモデルを作成する際にはＩＰソフトウェアモデルＬｉｂ（C言語，Ｃ＋＋言語又はアセンブラで記述）を利用する。

【００７７】次に、ステップＳＴ８では、ステップＳＴ６，ＳＴ７で作成したHWモデル及びSWモデルを利用して、HW／SW協調検証を行なうが、そのためにはHW－SW間のＩ／Ｆモデルが必要となる。そこで、以下の手順により、Ｉ／Ｆモデルモデルを作成する。

【００７８】まず、図１３に示すようなメモリマップを作成する。このメモリマップは、SWにプログラムをコンパイルしたものである。HWは、レジスタをメモリにマッピングする。

【００７９】そして、各機能ブロックのアドレスを決定し、各機能ブロックのアドレステーブルをデータバース化して、メモリマップＤＢに登録する。そして、図１４に示すように、メモリマップＤＢに登録されたHWのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する。

【００８０】バスコントローラは、各HWブロックのＰ（優先度）にしたがって、各バス毎のアービトレーション方法ごとに用意したＨＤＬテンプレートを生成する。すなわち、バスコントローラは、各ハードウェアのレジスタアドレスをメモリマップＤＢから抽出して、関数を作成し、これをドライバー関数テンプレートとしてSWモデルに格納する。

【００８１】図１５は、基本ハードウェアアクセス用のドライバー関数テンプレートの一例を示す図である。このようなメモリマップＤＢ，アドレスデコーダ，関数テンプレートを利用することにより、設計されたシステムのHWによってSWを動作させたり、システムのSWによってHWを動作させることが可能になり、HW／SW協調検証を行なうことができる。

【００８２】

【発明の効果】本発明により、ＩＰの機能に着目してより最適化された半導体集積回路装置を形成するための設計が可能になった。

【図面の簡単な説明】

【図１】（ａ），（ｂ），（ｃ）は、ＩＰ群を搭載するための配線基板となるシリコン配線基板の平面図、シリコン配線基板上に搭載されるＩＰ群の例を示す平面図、及びシリコン配線基板の断面図である。

【図２】本発明の実施形態におけるHW／SW協調設計の概略的な手順を示すフロー図である。

【図３】機能設計とＩＰ機能設計との詳細な処理の内容を説明するための部分フロー図である。

【図４】システムアーキテクチャ設計における処理の内容を説明するための部分フロー図である。

【図５】マッピングと性能解析の概略とを説明するための部分フロー図である。

【図６】各種のマッピングを行なった結果のコスト関数（Cost-tot）の値の例を示す図である。
【図７】静的なＨＷ／ＳＷ分割を説明するための図である。
【図８】（ａ），（ｂ）は、動的なＨＷ／ＳＷの切り換えにおける消費電力Ｐｏｗｅｒのタイムチャート図である。
【図９】消費電力を考慮したＨＷ／ＳＷの切り換え制御のためのプログラムの例をＣ言語で示す図である。
【図１０】（ａ），（ｂ）は、動的なＨＷ／ＳＷの切り換えにおける処理速度Ｓｐｅｅｄの消費電力依存性を示す図である。
【図１１】処理速度を考慮したＨＷ／ＳＷの切り換え制御のためのプログラムの例をＣ言語で示す図である。
【図１２】協調検証を説明するための図である。
【図１３】ＳＷにプログラムをコンパイルして得られたメモリマップを示す図である。
【図１４】メモリマップＤＢに登録されたＨＷのレジスタとアドレスとの対応表をとりだして、アドレスデコーダを作成する処理を示す図である。
【図１５】基本ハードウェアアクセス用のドライバー関数テンプレートの一例を示す図である。
【符号の説明】
１０　シリコン配線基板
１１　シリコン基板
１２　グランドプレーン
１３　第１配線層
１４　第２配線層
１５　パッド

---

フロントページの続き

| (51)Int.Cl.' | | 識別記号 | FI | | テーマコード'（参考） |
|---|---|---|---|---|---|
| Ｇ０６Ｆ | 17/50 | ６６４ | Ｇ０６Ｆ | 17/50 | ６６４Ａ |
| | | ６６６ | | | ６６６Ｔ |
| | | | | | ６６６Ｘ |
| | | ６６８ | | | ６６８Ａ |
| Ｈ０１Ｌ | 21/82 | | Ｈ０１Ｌ | 21/82 | Ｂ |
| | 27/04 | | | | Ｃ |
| | 21/822 | | | 27/04 | Ｕ |

Ｆターム(参考)　5B046 AA08 BA02 KA06
5F038 CA03 DF04 DF05 DF12 DF14
EZ08 EZ09 EZ10 EZ20
5F064 BB02 BB09 BB12 BB13 BB14
BB21 EE09 EE22 EE27 EE41
EE47 HH06 HH08 HH09 HH10
HH12